Animating Coupling between Inviscid Free-Surface Liquids and Elastic Deformable Bodies

by

Omar Zarifi

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, 2017

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

Driven by demand for high-fidelity computer-generated imagery, physics-based animation has become an exciting frontier of research in computer science. Simulations of fluids and their interactions with other objects in the environment have particularly enjoyed much attention and investigation. Consequently, effective techniques have been developed to efficiently simulate two-way coupling between fluids and rigid bodies, allowing for convincing animation of, for instance, ships on the ocean. On the other hand, accurately capturing interactions between fluids and deformable solids has proven to be much more elusive. In particular, satisfaction of boundary conditions poses a significant difficulty, as the straightforward voxelized treatment suffers from visible grid artefacts, whereas use of a conforming mesh greatly increases the computational overhead of a simulation.

This thesis investigates the problem of animating two-way coupling effects between free-surface liquids and linearly elastic solids. Aside from presenting simulation techniques for such liquids and solids separately, we introduce a new approach to simulating their interactions that exhibits several notable advantages over previous techniques. By fully incorporating the dynamics of the solid into pressure projection, we simultaneously handle fluid incompressibility and solid elasticity and damping. Thanks to this strong coupling, our method does not suffer from instability, even in very taxing scenarios. Furthermore, use of a cut-cell discretization methodology allows us to accurately apply proper free-slip boundary conditions at the exact solid-fluid interface. Consequently, our method is capable of correctly simulating inviscid tangential flow, devoid of grid artefacts or artificial sticking. Lastly, we present an efficient algebraic transformation to convert the indefinite coupled pressure projection system into positive-definite form. The thesis also contains an evaluation of our proposed method, including several animation scenarios, as well as comparisons to previous techniques.

Acknowledgments

I would like to thank my supervisor Christopher Batty for his excellent advice and support throughout the course my Master's program: my success in this endeavor is largely due to his patience and guidance. I would also like to thank my committee members, Gladimir Baranoski and Jeff Orchard, for taking the time to read my thesis and evaluate my work.

Residents of the Scientific Computing lab deserve considerable credit for making our office a pleasant and enjoyable workplace. Special thanks go to Parsiad, Eddie, and Colleen for introducing me to the Math Coffee and Donut shop and allowing me to partake in our refreshing coffee breaks.

I would also like to thank my colleagues at the Computational Motion Group: Jumyung, Ryan, Jade, Yipeng, Egor, and Dustin. Their colourful characters and our common interests ensured that our discussions were always amusing and insightful.

I am very grateful to all of the friends I have made in Waterloo and Toronto, especially Amit, Camila, Sajin, Savio, Raisa, and Nupur. I would be hopelessly bored outside of work hours without their companionship.

Finally, I would like to thank my family: my mother Nasrin, my father Abdul Hai, my sisters Nargis, Zohal, and Zarifa, and my beautiful nieces Sumaya, Leay, and Lily. Their unconditional love and support have fuelled me, not only during the course of this Master's program, but throughout my entire life. From the bottom of my heart—*thank you*...

Dedication

To my parents, Abdul Hai and Nasrin.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Fluids are an integral part of our everyday lives: from massive tidal waves crashing against eroding cliffs and thick plumes of smoke rising out of our chimneys to hot water slowly flowing past tiny solid particles and dripping through a porous filter into a coffee pot. Indeed, so familiar are we with how fluids behave that even relatively minor inaccuracies in their animation can be immersion-breaking. Consequently, artists must take great care when creating computer-generated scenes involving fluids. While early physics-based approaches effectively streamlined the animation process, they relied on simplified and idealized models that did not fully capture reality. In search of better simulation techniques, results from computational fluid dynamics have historically been applied for animation purposes. However, much of the research in the aforementioned field is aimed at engineering applications, where the primary objective is not visual fidelity, but accurate calculation of physical quantities (such as forces and stresses). In other words, simulation of fluids with the relatively new objective of high-quality visualization is a distinct and challenging problem, requiring and deserving thorough investigation. It is therefore not surprising that a vast amount of resources is dedicated to developing methods for realistic and efficient animation of fluids within the computer graphics community.

While the full dynamics of a fluid are governed by the famous Navier-Stokes equations, the simpler incompressible Euler equations can adequately capture the behaviour of many fluids in typical scenarios. In particular, these equations are sufficient for incompressible materials in cases where the viscous and thermal effects on the flow can be safely neglected. Note that many fluids do not exhibit noticeable viscosity effects and are macroscopically incompressible under ordinary conditions (e.g., water at room temperature). As a result, this simpler formulation can be used to convincingly animate a wide variety of scenarios, from ocean waves and flowing rivers to spilling milk and splashing rain.

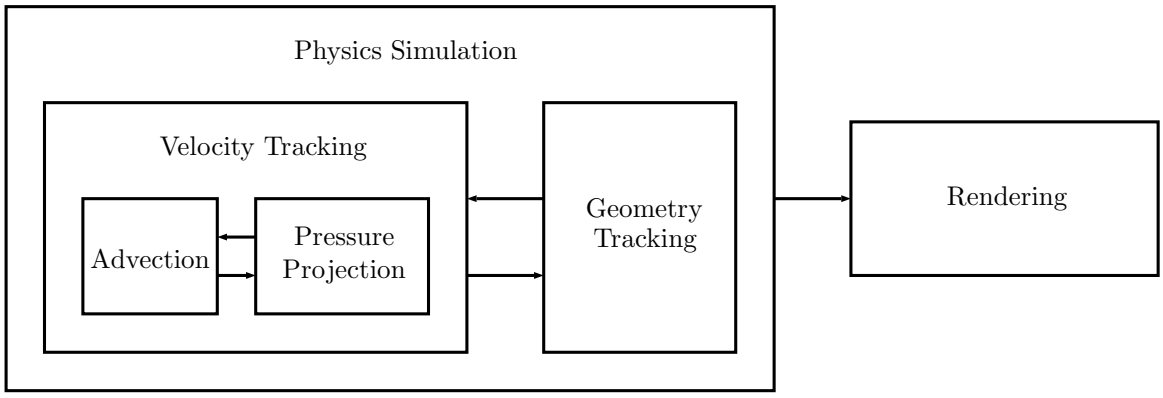

Figure 1.1: A typical (inviscid) fluid animation pipeline. Arrows indicate data flow.

In this thesis, we consider animation of inviscid free-surface liquids. Such fluids are characterized by the presence of the free-surface boundary, against which the liquid is allowed to push freely without experiencing any resistance. In essence, this assumption entails that the flow takes place within a vacuum; however, the free-surface also roughly models the interface between the liquid and air, which is present in a large proportion of animation scenarios. Although air itself is a fluid, it possesses a much lower density than common liquids (air and water densities, for instance, are three orders of magnitude apart). The tremendous difference in densities greatly limits the degree to which air can impact the liquid's behaviour. On the other hand, simulating two-phase flow with air treated as another fluid significantly raises computational cost and produces largely similar results (notable exception being preservation of bubbles, which rapidly collapse under free-surface conditions). These facts further justify the free-surface approximation and explain its prevalence in animation packages.

The fluid animation process is comprised of two decoupled components, namely the physics simulation and rendering (see Figure 1.1). The latter concerns itself with generating images to visualize the simulated data and is not within the scope of this thesis; instead, we are interested in simulation of the continuum mechanics. Many methods exist for tracking the liquid volume and surface for rendering purposes, offering various levels of simplicity, accuracy, speed, and robustness, and one may generally choose an algorithm freely. In order to track the liquid geometry, the simulator must also evolve the fluid's velocity field through time. Since liquids are macroscopically incompressible, our previous observations suggest that behaviour of inviscid free-surface liquids can be reasonably captured by the incompressible Euler equations. However, mathematical complexity of these

partial differential equations does not permit an analytic solution, requiring deployment of numerical methods. A common way to solve them in this manner relies on splitting the time step into two sequential stages: advection (which carries the velocity field along with the flow) and pressure projection (which enforces incompressibility and boundary conditions). The various components of a free-surface liquid simulator shall be discussed in more detail in Chapter 4; for now, we merely mention the importance of the pressure projection step in facilitating momentum exchange between the fluid and objects in contact with it. As such, this step plays a key role in solving the tackled problem of simulating interactions between fluids and deformable solids.

Although fluids and solids are similarly modelled as continuum matter, and their governing equations of motions are derived from Newton's laws of classical mechanics, their reaction to applied forces and deformations are nonetheless very different. An elastic solid, for example, will seek to undo any applied deformations to its shape. Consequently, one can define a rest configuration for such a solid; points that are disturbed from their reference location will then be subjected to internal forces as the body attempts to regain its rest form. This observation motivates the Lagrangian treatment of solids, where the points on the body are followed as the solid traces its trajectory through space. Note that following fluid particles in a similar fashion does not yield the same advantage, as a fluid's response to forces is not related to any rest state. In other words, knowing the location of a fluid point at a previous time reveals no information about the current mechanisms driving it. Hence, it is more natural to treat fluids in an Eulerian manner instead: the flow is analyzed with respect to a fixed frame of reference.

The focus of this thesis is coupling between inviscid free-surface liquids and linearly elastic deformable solids. The objective is reproduction of two-way coupling effects. That is, the liquid should be able to affect the motion of the solid through application of pressure forces, and, conversely, solid's presence should realistically deflect the flow. Since the liquid is not viscous, the free-slip condition must be enforced at the boundary separating these two entities. This condition prevents the liquid from entering the solid, but enables tangential flow to the surface. Lastly, a simulation algorithm must possess certain properties to be considered useful; based on animation-specific demands, we chose to prioritize stability, efficiency, and simplicity.

We have developed a novel coupling method to simulate interactions between free-surface liquids and linearly elastic deformable solids. The features of our coupling approach are outlined in Section 1.1, where we also reveal the degree to which the listed specifications are met. Our work is set to appear at the 2017 Symposium on Computer Animation [66], and much of the material that appears in this thesis draws from this publication.

1.1 Contributions

The subsequent chapters discuss the details of our coupling scheme; below, we outline its main features.

- **Excellent Stability.** Our simulation loop utilizes semi-Lagrangian advection in conjunction with backward Euler time integration, and, as a result, is extremely stable.
- **Boundary Condition Enforcement at the True Interface.** Use of cut-cell discretization allows us to apply the boundary conditions at their exact location, even if the solid does not line up with the fluid grid.
- **Tangential Free-Slip without Grid Artefacts.** Our coupling method constrains only the normal component of fluid velocity at the boundary. Thus, the liquid may move tangentially to the solid, as required of inviscid flow.
- **A Symmetric Positive-Definite Linear System.** The described pressure projection scheme requires the solution of a symmetric positive-definite system, without requiring factorization of the solid damping or stiffness matrices.
- **No Extra Constraints or Degrees of Freedom.** As opposed to some previous methods, we do not introduce any Lagrange multipliers in order to satisfy boundary conditions. That is, the number of variables in the linear system is exactly equal to the number of relevant fluid pressure samples plus the number of solid degrees of freedom.

1.2 Outline

The purpose of the present chapter was to describe and motivate the problem of two-way solid-fluid coupling and explicitly list our contributions. The remainder of the thesis is organized as follows. Chapter 2 provides a brief survey of previous works that are relevant to the problem at hand. Chapters 3 and 4 introduce the governing equations of motion for solids and fluids, respectively; computational techniques to simulate objects of each type are also included. Our novel contribution, the solid-fluid coupling scheme, is described in Chapter 5. Chapter 6 is dedicated to evaluation of our method, while concluding remarks and possible avenues for future research are contained in Chapter 7.

Chapter 2

Related Works

This project is naturally built upon and motivated by a myriad of previous works; this chapter is dedicated to summarizing these relevant parts of the literature. Historical techniques for separate animation of liquids and solids are first briefly outlined, followed by a more in-depth survey of approaches to couple these entities.

2.1 Liquid Animation

We begin with a review of early literature pertaining to animation of liquids. Prevalence of fluids in everyday life makes them an important target for visual effects, and, despite the state of computer hardware in 1980's, this problem was already being considered. Yet the computational limitations could not be practically ignored; as such, earlier methods did not aim to solve the physical dynamics equations, but opted for imitating realistic surface behaviour through blending of waves [24, 39]. Another popular approach involved the use of height field representation to approximately solve the shallow water equations [33]. This technique was subsequently improved by Chen and Lobo [12], who considered the two-dimensional Navier-Stokes equations instead. A full three-dimensional treatment of these governing equations was provided by Foster and Metaxas [23], whose work also popularized use of the marker-and-cell grid for animation of fluids, three decades after this discretization scheme was first proposed by Harlow and Welch [30]. Stam [58] proposed an improvement to this simulation technique for smoke in the form of unconditionally stable semi-Lagrangian advection scheme. This approach was finally adopted for the purpose of liquid simulations by Foster and Fedkiw [22]. In addition, Foster and Fedkiw [22] pioneered the use of level set surface tracking, which has since been commonly used in

conjunction with particles for efficient re-creation of dynamic liquid behaviour, such as splashing [17, 18, 19].

2.2 Solid Animation

The groundwork for physics-based animation of elastic solids was laid by Terzopoulos et al. [63], who defined deformation potential energy functions and used a finite difference methodology for their discrete formulation. Terzopoulos and Fleischer [62] subsequently modified the discretization scheme to instead utilize a volumetric finite element mesh, while keeping the core strategy unchanged. Although these seminal papers made use of a semi-implicit integration scheme to solve the dynamics equations, much of the following work relied on simpler, but less stable, explicit methods (for example, [8, 44]). Citing stiffness of the arising differential equations, Baraff and Witkin [3] advocated for use of implicit schemes, deploying linearization of the forces to simplify integration. A similar strategy is to perform linearization at the model level to allow for fully implicit treatment of the discrete equations; however, the straightforward way to do so suffers from rotational artefacts. This issue was effectively avoided by Etzmuß et al. [20] in cloth simulation with use of a corotational model; subsequently, Müller and Gross [42] applied this technique to simulate volumetric solids. Noting its advantages, we also used this model to animate linearly elastic solids for the purpose of this project.

2.3 Coupling Fluids to Rigid Bodies

We now proceed to survey techniques for the related problem of animating interactions between fluids and rigid bodies. Since a typical fluid simulation often involves an enclosing container, the problem of rigid boundaries and objects has been considered from the earliest days of fluid animation. Use of the now-ubiquitous marker-and-cell discretization was introduced to the computer graphics community by Foster and Metaxas [23]. In this seminal paper, grid-aligned solid boundaries were readily handled; however, significant voxelization artefacts unavoidably cropped up in the presence of sloped or curved geometry. Furthermore, only a limited discussion of dynamic rigid bodies was included, with the authors remarking that a more sophisticated method is necessary to properly capture two-way coupling.

Handling of solid boundary conditions, particularly for advection, was subsequently improved by Foster and Fedkiw [22], Houston et al. [32], and Rasmussen et al. [46]. How-

ever, issues could still surface due to the voxelized pressure solve. For instance, none of the aforementioned methods can adequately simulate a body of water in hydrostatic equilibrium within a spherical container. Feldman et al. [21] circumvented this problem by using conforming volumetric meshes in order to accurately capture the fluid region (and, hence, its boundaries), and Klingner et al. [34] adapted this approach for the purpose of two-way coupling with dynamic rigid bodies. Unfortunately, both of these works relied on the use of conforming unstructured tetrahedral meshes and frequent remeshing of the fluid domain, adding computational overhead. Another approach to two-way coupling is the "rigid fluid" method proposed by Carlson et al. [11], which handles interactions by temporarily treating the rigid body as a fluid region, but this approach can lead to severe leakage through objects.

A widely adopted solution to the problem of irregular solids is the use of cut-cell methods, which essentially clip the object geometry against a regular background grid. Their use in computer graphics was first suggested by Roble et al. [50], who proposed a simple modification to regular pressure projection that allows for more accurate enforcement of static solid boundary conditions. The variational formulation of Batty et al. [5] similarly treated two-way rigid body interactions by casting the coupled pressure solve in an energy minimization form that accounts for partial cell volumes in three dimensions. Subsequently, Ng et al. [43] used a finite volume cut-cell discretization to derive a more accurate scheme for kinematically scripted solids, involving the same stencils with a different choice of weighting terms. This thread of research culminated in work by Gibou and Min [27], who presented further accuracy improvements to the method of Ng et al. [43] and extended it to two-way coupling with dynamic rigid bodies.

Such cut-cell discretizations have become increasingly common in fluid animation, for example in the context of spatial adaptivity [6], multigrid methods [64], detailed splashes [16], and thin solids [2]. This trend can be attributed to the simplicity of cut-cells (both conceptual and with respect to implementation) as well as their expressive power: fluxes across small geometrical details can be mathematically captured without refining or re-orienting the computational grid. Observing these properties, our coupling scheme makes use of cut-cell discretization to accurately enforce coupling conditions at the interface between a fluid and a deformable body.

2.4 Coupling Fluids to Lagrangian Deformables

The fluid nature of liquids and gases results in immense shape changes throughout the course of a simulation. Consequently, despite various advantages possessed by Lagrangian

mesh-based approaches to fluid simulation [14, 40], such as a unified physical model for solids and fluids, better volume preservation, and straightforward support for implicit surface tension, they are less common because they necessitate continuous and expensive remeshing of the fluid region. Many solids, on the other hand, are not ordinarily subject to such extreme distortion and permanent deformation. As a result, Lagrangian mesh methods have predominantly been used to animate elastic deformable objects. A typical example of such a scheme is given by Teran et al. [61]: the solid domain is partitioned into volumetric elements, allowing for a simple calculation of per-element strains, which can be used to obtain material stresses and internal forces.

This discrepancy in representation of fluids and solids introduces a challenge in their dynamic coupling: because velocity samples for the two entities are not collocated, it becomes harder to accurately enforce boundary conditions at the regions of contact. To bypass the inherent difficulty in simultaneous satisfaction of boundary conditions, early attempts at animating solid-fluid interactions took a weakly coupled approach that alternates solving for solid and fluid motions. For example, Genevaux et al. [26] devised a simple simulation scheme that integrates fluid and solid dynamics separately, with interactions facilitated by interfacial forces. Guendelman et al. [28] took a similar approach, but used solid velocities as prescribed boundary conditions for pressure projection. Effects of the fluid on the solid are then captured through application of forces induced by the computed pressure field. We observe that the simplicity of weakly coupled schemes generally comes at the cost of accuracy, stability, or efficiency: it is often necessary to take restrictively small time steps in order to prevent unphysical behavior or even catastrophic simulation failure.

The problem of instability was largely mitigated by Chentanez et al. [13], whose implicit, strongly coupled formulation is demonstrably robust, even with relatively large time steps. The tetrahedral mesh variant of this method naturally requires continuous remeshing, while the regular grid variant does not correctly apply the free-slip condition at the exact interface, since it inaccurately treats the solid as voxelized; as a result, visible voxelization artefacts appear at the solid-fluid boundary. Furthermore, the linear system for pressure projection is non-symmetric, rendering it more difficult to solve numerically. The formulation of Robinson-Mosher et al. [49], on the other hand, instead requires the solution of a symmetric indefinite matrix. While such systems are generally easier to solve numerically than non-symmetric ones, working with positive-definite matrices is still preferred. The effects of the underlying voxelization are present in this method as well; in particular, lumping of fluid and solid momenta within computational cells forces the fluid to inherit the solid's velocity at the boundaries. In essence, despite the use of an inviscid fluid model, this method violates the desired free-slip condition, causing the fluid to stick unnaturally to objects.

In subsequent follow-up work, Robinson-Mosher et al. [47] incorporated tangential free-slip boundary conditions into their coupled pressure projection scheme via Lagrange multipliers that explicitly constrain only the interpolated normal component of relative velocity to be zero. However, these boundary conditions are once again applied at regular grid faces as opposed to the true contact interface, and the resultant linear system remains symmetric indefinite. A positive-definite formulation of the same problem was finally presented by Robinson-Mosher et al. [48] through linear algebraic manipulations that rely on construction of a symmetric factorization of the solid's damping matrix; our method is more general in avoiding this requirement. In addition to our use of a cut-cell approach that better accounts for solid geometry, our work is also distinguished from the preceding three approaches by our use of corotational linear elasticity which allows us to incorporate the full implicit solid dynamics into the coupled pressure projection; Robinson-Mosher et al. include only the solid damping.

An important determinant of computational cost of a deformable simulation is the node count for the mesh. In certain cases, it is not possible to coarsen a mesh without significantly affecting its shape. In these scenarios, one can still improve running time by utilizing a reduced model, whose deformations are restricted to a smaller subset (e.g., [53]). Building on the aforementioned algorithm of Robinson-Mosher et al. [48], Lu et al. [37] developed a method to capture interactions of fluids and reduced deformable bodies. Since our goal is high-fidelity simulation with accurate boundary condition handling, we do not consider reduced models; however, we anticipate that our coupling strategy could similarly be adapted to work with reduced models.

Sotiropoulos and Yang [57] provide a review of a wide variety of approaches to fluid-structure interaction in computational physics. While various flavors of immersed boundary and cut-cell methodologies are common, we are not aware of any method that offers the specific advantages provided by our scheme. As an example, Pasquariello et al. [45] use a cut-cell finite difference/volume strategy in a manner broadly similar to our work and that of Ng et al. [43]. In contrast to our work though, they consider compressible flow, handle the boundary interactions via a mortar method based on explicit Lagrange multiplier constraints, and employ a weak/partitioned coupling strategy.

2.5 Coupling Fluids to Eulerian Deformables

Although still not as popular as the ubiquitous Lagrangian schemes, Eulerian and hybrid solid simulation approaches have recently gained traction; the Eulerian solids approach [35]

and the material point method [59] are representative examples. Most relevant to our research is the Eulerian two-way solid-fluid coupling method of Teng et al. [60]. While a fully Eulerian scheme is in many ways very natural, this method exhibits several practical drawbacks. First, the underlying Eulerian solid model suffers from strong numerical damping. More importantly, this unified model necessarily implies a no-slip condition at the boundaries, and the described method is limited to simulating fully immersed, strictly incompressible objects. By contrast, we are interested in coupling of potentially compressible solids to inviscid free-surface liquids, for which the free-slip condition yields more realistic results for animation scenarios.

2.6 Smoothed Particle Hydrodynamics Methods

Smoothed particle hydrodynamics (SPH) methods provide an alternate way to simulate fluids, by distributing mass to particles and using interactions between these constituents to determine the dynamics of the aggregate fluid [7, 15, 41]. Several algorithms to simulate two-way coupling of deformable solids to SPH fluids have been previously proposed (e.g., [1, 56]). These methods are outside the scope of this thesis, which focuses on Eulerian fluid representations.

Chapter 3

Solid Dynamics

Imagine a solid object at rest in a total vacuum, without any nearby agents that may influence its motion. In such a rest state, the object occupies some space (say Ω), and, as governed by Newton's first law of motion, it shall remain in this rest state until acted upon by an external force.

Suppose now that a force is applied to some region R of Ω. According to Newton's second law of motion, an acceleration is induced on R, proportional to the magnitude of the force. But R cannot move independently, since it is coupled to its neighbours in Ω. That is, despite local action of the force, all parts of the solid are affected. To determine how the aggregate body, initially occupying Ω, evolves in time, a model is needed to describe the internal coupling within the solid.

The simplest such model treats solids as ideal rigid bodies. The underlying assumption of this model is that the distance between any two points within the object remains constant, regardless of what happens to the solid. Accepting this axiom, one can show that the state of the object at any given time can be captured by two quantities: its position and its orientation. In three dimensions, both of these properties can be represented by three-dimensional vectors. Hence, the configuration of a three-dimensional rigid body can be summarized by a mere six numbers. In many situations, the described model is an excellent approximation and its simplicity makes it an attractive choice for simulation purposes. For instance, a billiard ball will maintain its spherical shape after impact with the cue stick and collisions with other balls. Likewise, a steel ball bearing will not dent or crack under any amount of pressure a human may manually subject it to.

However, the rigid body model is admittedly inadequate for capturing the behaviour of softer solids, like a rubber ball or a jellied dessert. We return to our example of local

force application to illustrate the conceptual difference in dynamics. For a rigid body, the resultant acceleration at R immediately affects the entire object, since any delay in reaction will inflict a change in distances between points of the solid. If the solid is not rigid, on the other hand, then the portion within R will start to move independently, modifying the object's shape. In response to this deformation, internal forces will be generated; it is these forces that gradually cause the rest of the body to move.

It is clear that a general three-dimensional solid possesses many more degrees of freedom than a rigid body. This fact raises the question: how can the configuration of a deformable body be described? Furthermore, how can we measure the deviation of an arbitrary configuration from a more stable rest state? Finally, given this deviation, what kinds of internal forces are generated by the material in a drive towards a more stable configuration? This chapter is dedicated to mathematically answering these questions in a continuous framework, as well as providing a discretization scheme for simulation purposes. We note that much of this material can be found within standard sources on the subject, such as the book by Segel and Handelman [52] and course notes by Sifakis and Barbic [54].

3.1 Deformations

Recall the previous example of an isolated, stationary object. It is said that this solid is in a rest state, since its configuration does not cause internal forces to be generated. If we freeze the object at a later time t, after it has been influenced by external stimuli and undergone motion and deformation, we find that it occupies a different region, Ω_t. Although Ω_t contains some information about the solid, it is not an adequate description of the object's physical state. For instance, consider the scenario depicted in Figure 3.1; it is impossible to deduce from Ω_t whether the solid was deformed or simply rotated. Yet these two circumstances clearly differ: internal forces will arise only in the former case, as the solid is still in a rest state if it merely undergoes a rotation.

A much more expressive description of physical configuration is the deformation map. Towards defining this notion, we follow the trajectory of a specific solid "particle", initially at $\mathbf{X} \in \Omega$, as the body undergoes its motion; in the current configuration, we find that this particle has moved to some position $\mathbf{x} \in \Omega_t$. If we track the motion of every solid constituent in this manner, we can clearly obtain the shape of the entire object. In addition to the shape, however, this scheme for describing configuration also reveals the change in internal structure of the solid. Revisiting the previously ambiguous example of an ellipse, Figure 3.2 illustrates the expressive power of particle tracking: we can effortlessly distinguish

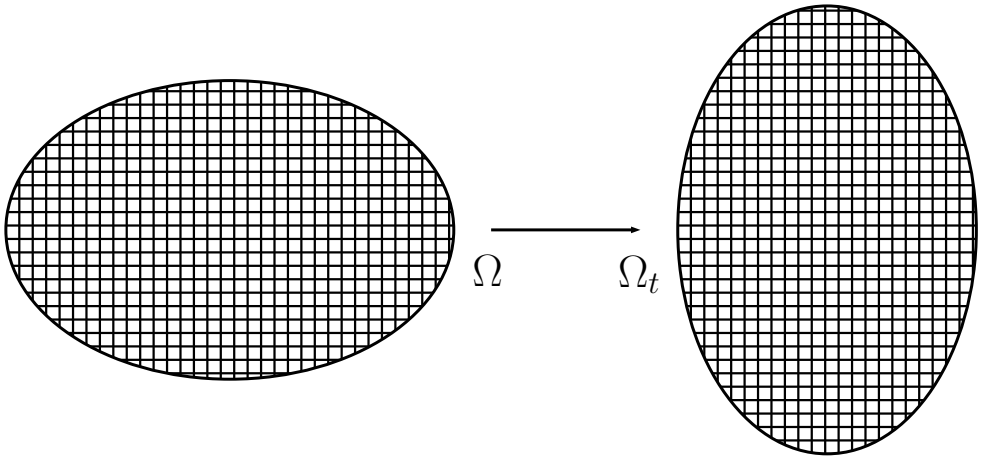

Figure 3.1: Evolution of a solid; it is impossible to tell if the object was deformed or rotated by comparing occupied regions.

between a deformation from a rotation by comparing relative particle locations to the reference configuration.

The deformation map is a mathematical construct that allows us to capture the change in distribution of matter within a solid. Formally, it is a function $\phi : \Omega \to \Omega_t$, such that $\phi(\mathbf{X})$ is the current location of the particle that was at $\mathbf{X}$ in the rest state. For example, if the solid is simply translated, then its deformation map will have the closed form $\phi(\mathbf{X}) = \mathbf{X} + \mathbf{t}$ for some vector $\mathbf{t}$; similarly, if the solid is stretched uniformly in every direction by a factor of a, then its deformation map can be expressed as $\phi(\mathbf{X}) = a\mathbf{X}$. Thus, the deformation map captures the state of every material particle in a mathematically succinct manner; but what is a "particle"? For simulation purposes, a particle could simply represent a small chunk of the object. In the physical limit, it could denote a molecule or an atom of the comprising matter. In both of these cases, Ω and Ω_t would be finite sets of positions, equal in cardinality to the number of particles in the solid. However, it is more mathematically convenient to consider these sets as connected regions in $\mathbb{R}^3$ (and, consequently, particles to be points in space), allowing use of vector calculus as an analytic tool. For example, this perspective enables us to differentiate the deformation map in space to obtain the deformation gradient, an important indicator of type and severity of shape change. By treating solids in this manner, we adopt the viewpoint that all of the space inside the object is continuously filled with matter; this is the underlying assumption of continuum mechanics. Although this is fundamentally at odds with reality,

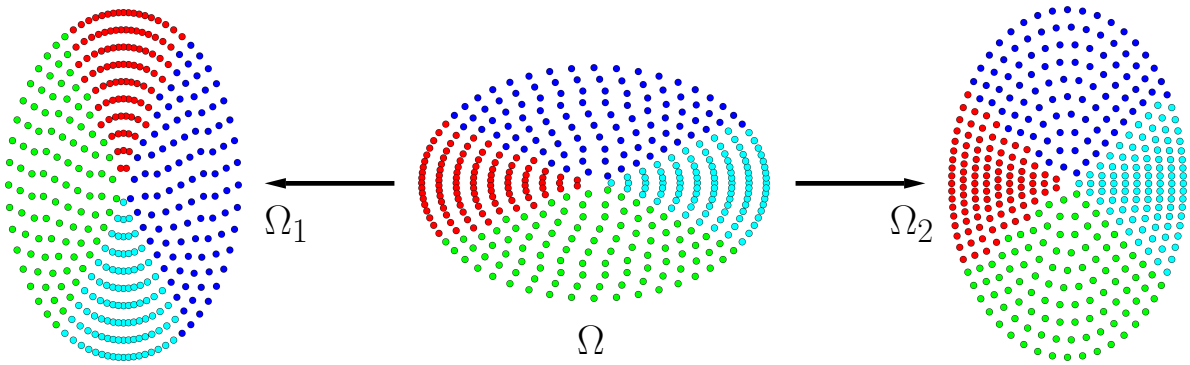

Figure 3.2: Two different configurations of a solid with the same shape. Note how relative positions of particles changed in Ω_2, signalling deformation.

it is nevertheless an excellent approximation at macroscopic scales.

3.2 Strain and Stress

This section explains the physical consequences of induced shape changes. We begin with discussion of strain, which quantifies the extent of deformation for a given state, then continue on to define stress, which summarizes the internal forces that arise in reaction to strain. Link between stress and strain is provided by the constitutive model for the considered materials, description of which concludes this section.

3.2.1 Strain

For a given configuration of the solid, strain is a measure of its deviation from a rest state. As such, it can be considered a function of the deformation map; since different parts of the body can experience different degrees of distortion, it must also be a function of location. Thus, we can denote it by $S(\phi, \mathbf{X})$. Towards simplifying the dependence on the deformation map, we can note that strain at a material point $\mathbf{X}_0$ within the solid is contingent on the configuration of its local neighbourhood; within such a local neighbourhood, linearization of the deformation map yields a good approximation:

$$\phi(\mathbf{X}) \approx \phi(\mathbf{X}_0) + \left.\frac{\partial \phi}{\partial \mathbf{X}}\right|_{\mathbf{X}=\mathbf{X}_0} (\mathbf{X} - \mathbf{X}_0) = \left.\frac{\partial \phi}{\partial \mathbf{X}}\right|_{\mathbf{X}=\mathbf{X}_0} \mathbf{X} + \left(\phi(\mathbf{X}_0) - \left.\frac{\partial \phi}{\partial \mathbf{X}}\right|_{\mathbf{X}=\mathbf{X}_0} \mathbf{X}_0\right). \quad (3.1)$$

Letting F denote the spatial derivative of ϕ and $\mathbf{t}(\mathbf{X}_0)$ denote the vector in the parentheses, we can write the above expression as $\phi(\mathbf{X}) \approx F(\mathbf{X}_0)\mathbf{X} + \mathbf{t}(\mathbf{X}_0)$. With respect to describing local deformation in the neighbourhood of $\mathbf{X}_0$, the vector $\mathbf{t}(\mathbf{X}_0)$ holds no value, as it corresponds to a rigid translation. Consequently, we conclude that strain is simply a function of the spatial derivative of ϕ; for its importance in determining strain, this derivative deserves its own name: the deformation gradient.

Having argued that strain depends on the deformation gradient, we now investigate this relationship more closely. As remarked previously, differentiation of the deformation map already filters out translations of the solid; a reasonable strain measure must likewise be invariant under the second type of rigid transformation, namely rotation. Suppose that the solid is rotated about the point $\mathbf{Y}$; then its deformation map will be of the form $\phi(\mathbf{X}) = R(\mathbf{X} - \mathbf{Y}) + \mathbf{Y}$ for some rotation matrix R. Clearly the resultant deformation gradient is simply $F = R$. Observing that rotation matrices are orthogonal, we can state that $F^T F = I$, where I denotes the identity matrix. Thus, we may consider it reasonable to take deviation of $F^T F$ from the identity matrix into account when measuring strain; in fact, the Green strain tensor does exactly this:

$$G(F) = \frac{1}{2}\left(F^T F - I\right). \tag{3.2}$$

Although Green strain is a powerful determinant of deformation severity, it is quadratic in the deformation gradient. As a result, combining use of Green strain with an implicit time integrator necessitates solution of a nonlinear system of equations at each time step. To alleviate this issue, linearization of the Green strain tensor (known as the small strain tensor) is often employed:

$$\epsilon(F) = \frac{1}{2}\left(F + F^T\right) - I. \tag{3.3}$$

Use of this strain measure may give rise to unrealistic behaviour if the simulated solid undergoes large deformations. For instance, even rotations yield non-zero small strain tensors. However, (3.3) is a good approximation to the Green strain for small deformations, and its linearity makes it an appealing choice for animation purposes. Furthermore, it is not difficult to eliminate the effect of rotations on internal dynamics, though we delay the discussion of such a scheme to Section 3.6.

3.2.2 Stress

Although strain allows us to assess the deviation of a configuration from a rest state, it does not directly reveal the nature of internal forces generated by the material. For this purpose, stress is defined.

As mentioned previously, effects of an externally applied force are propagated throughout the solid "sequentially". That is, the parts that experience the force directly begin to move; in turn, these parts influence their neighbours and so on. Transfer of momentum within the solid in this manner is modelled through contact forces between neighbouring chunks. To study these forces, we consider two neighbouring parts of a solid, denoted by C_1 and C_2. Effects of a deformation experienced by C_1 are transferred to C_2 via a force applied at their common interface; call this interface Γ and the force $\mathbf{F}$. It is clear that $\mathbf{F}$ is a function of how C_1 is pushing on C_2, as well as the area of contact. In particular, if we define $\mathbf{T}(\mathbf{x})$ to be force field per unit area exerted by C_1 on C_2 at the interface, then

$$\mathbf{F} = \iint_{\Gamma} \mathbf{T}(\mathbf{x}) dA. \tag{3.4}$$

As such, to complete our model of the solid's internal forces, we need only to describe the field $\mathbf{T}(\mathbf{x})$.

Contact force per unit area (the vector $\mathbf{T}$) is known as traction in solid dynamics. The mathematical nature of tractions is described by two results: Cauchy's postulate and Cauchy's stress theorem. The former asserts that traction at a point $\mathbf{x}$ depends only on the normal direction to the interface; thus, traction is a function of location within the solid, as well as the normal direction to the considered boundary: $\mathbf{T}(\mathbf{x}, \mathbf{n})$. Cauchy's stress theorem states that this function is linear in $\mathbf{n}$ for a fixed $\mathbf{x}$. Therefore, for each location within the solid, there exists a matrix $\sigma(\mathbf{x})$ such that $\mathbf{T}(\mathbf{x}, \mathbf{n}) = \sigma(\mathbf{x})\mathbf{n}$. Assuming $\mathbf{n}$ is the normal in the deformed configuration, σ is known as the Cauchy stress tensor. It effectively summarizes the state of internal forces within the object and, unsurprisingly, is an important part of solid simulation.

As an example, we may conceptually divide a virtual solid into tetrahedra $P_1, \ldots, P_n$; the four faces of P_i will be denoted by $f_i^1, f_i^2, f_i^3, f_i^4$. Assuming that the Cauchy stress tensor field can be computed, the net effect of internal forces on each part P_i is

$$\mathbf{F}_i = \iint_{\partial P_i} \sigma \mathbf{n} dA \approx \sum_{j=1}^{4} A(f_i^j)\sigma(\mathbf{c}(f_i^j))\mathbf{n}(f_i^j), \tag{3.5}$$

where $A(f), \mathbf{c}(f), \mathbf{n}(f)$ are the area, centre, and outward unit normal of face f. A simple simulation scheme can then be devised that uses these forces to alter the momentum of each tetrahedron. But this method requires recalculation of the face areas, centres, and normal vectors at each simulation step. The simulation scheme could be simplified if the forces on a part could be related to its state in the static undeformed configuration. To this end, the first Piola-Kirchoff stress tensor is defined.

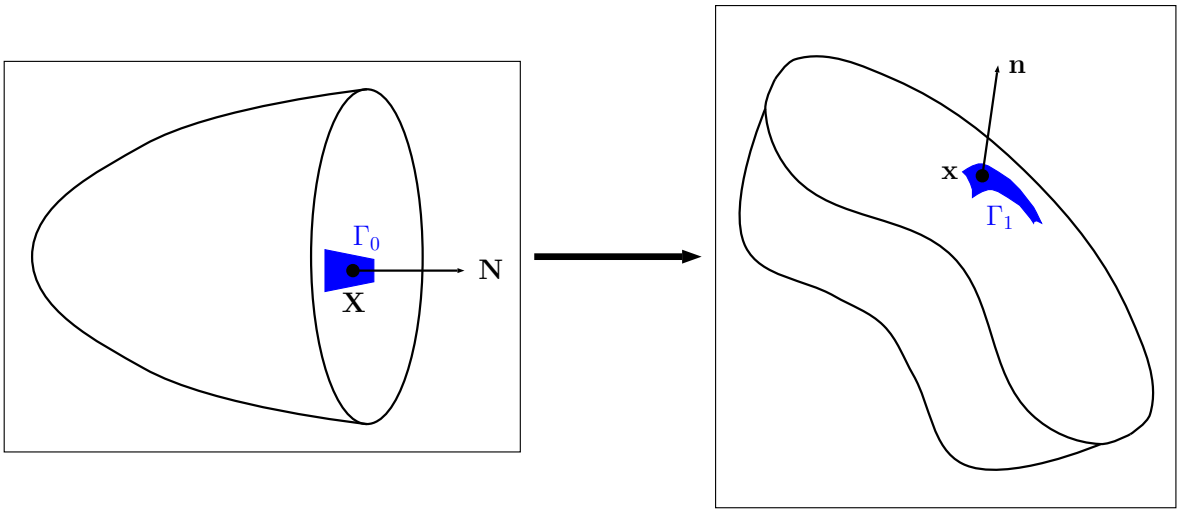

Figure 3.3: Transformation of a planar patch (generated by hypothetically cutting the solid with a plane).

The first Piola-Kirchoff stress tensor is written as P and its relationship to the Cauchy stress tensor is given by

$$P = \det(F)\sigma F^{-T}. \tag{3.6}$$

Utility of this stress measure lies in its ability to connect tractions within a body in the deformed state to normals in the rest configuration. Suppose that the body is cut in its rest state by a plane perpendicular to $\mathbf{N}$ going through $\mathbf{X}$ (see Figure 3.3). Let Γ_0 be some neighbourhood of $\mathbf{X}$ on the cut surface. After the (uncut) body has undergone some deformation (described by ϕ), the point at $\mathbf{X}$ has moved to $\mathbf{x}$, and Γ_0 has evolved into Γ_1. Letting A_1 be the area of Γ_1, internal force at this neighbourhood is approximately $A_1\sigma(\mathbf{x})\mathbf{n}$. On the other hand, the first Piola-Kirchoff stress tensor allows us to express this force as $A_0P(\mathbf{X})\mathbf{N}$, where A_0 is area of Γ_0. In continuous terms, we have

$$\iint_{\Gamma_1} \sigma d\mathbf{n} = \iint_{\Gamma_0} P d\mathbf{N}. \tag{3.7}$$

It is not difficult to see how this stress measure simplifies simulation schemes: assuming the first Piola-Kirchoff stress tensor can be calculated for a given state of the solid, the resultant internal forces are obtainable from geometry in the reference configuration. No need to track the evolution of normals and areas!

3.2.3 Material Model

Having explained the stress tensor conceptually, we next provide the final link between a body's configuration and the resultant internal forces: the stress-strain relationship. Note that different objects of the same shape can react differently to the same deformation (and, hence, the same strain). For instance, a stretched rubber sheet will rapidly spring back to its original configuration, but a distorted metal plate will happily maintain its newly-endowed shape. Some objects (such as a carbon fibre sheet) may even react differently to the same amount of induced deformation in different directions. Assuming that the same laws of motion apply to all these objects, we are forced to conclude that the relationship between stress and strain is governed by the material; as such, a material model is required.

In this thesis, we consider only isotropic linear hyperelastic objects. A material is said to be isotropic if it reacts the same way to deformations in every direction; so, a homogeneous rubber ball would be considered isotropic, whereas the aforementioned carbon fibre sheet is clearly not. Hyperelasticity implies that the body's internal forces always drive it towards its original rest state, devoid of hysteresis. Lastly, linearity of the model refers to the relationship between generated forces and the deformation gradient. Despite its seemingly limiting simplifying assumptions, this material model can be used to convincingly animate a large class of deformables, including inflatable toys, elastic bands, and even human muscles.

To satisfy linearity, we use the linearization of Green strain from (3.3). Given the assumptions about the simulation subject, it turns out that the simplest physically-sound constitutive equation is

$$P = 2\mu\epsilon + \lambda \operatorname{tr}(\epsilon) I, \tag{3.8}$$

where I is the identity matrix. As such, an isotropic linear hyperelastic material is characterized by two numbers: the constants λ and μ are known as the first and second Lamé parameters, respectively (note that the second Lamé parameter is also referred to as the shear modulus). They are physical properties of the material, related to Young's modulus (E, which measures forces that arise as a result of stretching) and Poisson's ratio (ν, which describes the degree of compressibility for the material) via

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \tag{3.9}$$

$$\mu = \frac{E}{2(1+\nu)}. \tag{3.10}$$

Recall that hyperelasticity implies that shape changes of the solid give rise to internal forces that work to undo them, regardless of how the shape changes came about. Equiva-

lently, a hyperelastic material conservatively stores the effects of deformation as potential energy, and internal forces are generated to push the solid into a state that minimizes this energy. Since it is not necessary for the entire body to undergo uniform deformation, not all parts of the solid will have the same amount of accumulated potential energy. As such, it is more natural to work with energy density, which is allowed to vary throughout the domain, depending on the severity of local deformation; we will denote this density field by $\Psi(F)$. Then the amount of potential energy stored in a part of the solid that initially occupied Σ is given by

$$E(\Sigma) = \iiint_{\Sigma} \Psi(F) dV. \tag{3.11}$$

For the constitutive relationship given in (3.8), the potential density function is

$$\Psi = \mu \operatorname{tr}(\epsilon^T \epsilon) + \frac{\lambda}{2} \operatorname{tr}^2(\epsilon). \tag{3.12}$$

A hyperelastic material has the additional useful property that the first Piola-Kirchoff stress tensor is simply the derivative of the potential density function with respect to the deformation gradient:

$$P = \frac{\partial \Psi}{\partial F}. \tag{3.13}$$

We will exploit these facts about hyperelasticity during discretization.

3.3 Equations of Motion

Stress within a solid is governed by its configuration; in turn, the configuration evolves based on the stresses within the solid. The manner by which these quantities simultaneously evolve is governed by a set of partial differential equations. Towards deriving these equations, consider a small part of the solid, occupying Σ in the rest state. In the current configuration, this part is assumed to be subject to gravity, as well as internal forces; so, total force experienced by this part is

$$\mathbf{f}_{\Sigma} = \iiint_{\Sigma} \rho \mathbf{g} dV + \iint_{\partial\Sigma} \mathbf{T} dA = \iiint_{\Sigma} \rho \mathbf{g} dV + \iint_{\partial\Sigma} P d\mathbf{N}, \tag{3.14}$$

where $\mathbf{g}$ is acceleration due to gravity and ρ is the mass density field of the solid (with respect to the reference configuration). Invoking the divergence theorem, these forces can be expressed as

$$\mathbf{f}_{\Sigma} = \iiint_{\Sigma} \rho \mathbf{g} dV + \iiint_{\Sigma} \nabla \cdot P^T dV = \iiint_{\Sigma} \left(\rho \mathbf{g} + \nabla \cdot P^T \right) dV. \tag{3.15}$$

On the other hand, Newton's second law states that rate of change of momentum with respect to time is equal to force. Thus, if we let $\mathbf{v}$ denote the Lagrangian velocity field, then

$$\frac{d}{dt}\iiint_{\Sigma}\rho\mathbf{v}dV=\mathbf{f}_{\Sigma}. \tag{3.16}$$

Equivalently, we have

$$\iiint_{\Sigma}\rho\frac{\partial\mathbf{v}}{\partial t}dV=\iiint_{\Sigma}\left(\rho\mathbf{g}+\nabla\cdot P^{T}\right)dV. \tag{3.17}$$

Since this equality holds for all Σ, we can conclude that

$$\rho\dot{\mathbf{v}}=\rho\mathbf{g}+\nabla\cdot P^{T}. \tag{3.18}$$

These are the equations of motion that we wish to solve. To avoid confusion, we emphasize that matrix divergence is applied to the columns, and the derivatives are taken with respect to the reference coordinates.

3.4 Variational Treatment of Elasticity

We split discretization of the partial differential equations into two parts: time and spatial. We will discuss the latter in Section 3.5; for now we focus on discretizing time by casting the problem as an optimization.

Consider a solid, initially occupying Ω. We claim that solving

$$\arg\min_{\mathbf{v}}\iiint_{\Omega}\left(\frac{1}{2}\rho(\mathbf{v}-\mathbf{v}^{*})\cdot(\mathbf{v}-\mathbf{v}^{*})+\Psi(F^{*}+\Delta t\nabla\mathbf{v})\right)dV \tag{3.19}$$

is equivalent to discretely advancing the velocity field by a time step of Δt. In this expression, $\mathbf{v}$ is the velocity field, and asterisks denote quantities before the time step; to simplify our analysis, we assume without a loss of generality that effects of gravity are already incorporated into $\mathbf{v}^{*}$.

In order to justify the claim, we let $\mathcal{F}(\mathbf{v})$ denote the functional that is optimized in (3.19). Suppose that it achieves its minimum for some velocity field $\mathbf{v}$ and consider $h(\delta)=\mathcal{F}(\mathbf{v}+\delta\mathbf{w})$, where $\delta\in\mathbb{R}$ and $\mathbf{w}$ is some smooth function:

$$\begin{aligned}h(\delta)=&\iiint_{\Omega}\frac{1}{2}\rho\left((\mathbf{v}-\mathbf{v}^{*})\cdot(\mathbf{v}-\mathbf{v}^{*})+2\delta(\mathbf{v}-\mathbf{v}^{*})\cdot\mathbf{w}+\delta^{2}\mathbf{w}\cdot\mathbf{w}\right)dV\\&+\iiint_{\Omega}\Psi(F^{*}+\Delta t\nabla\mathbf{v}+\delta\Delta t\nabla\mathbf{w})dV.\end{aligned} \tag{3.20}$$

Recall that $\mathcal{F}$ has a minimum at $\mathbf{v}$; it follows that h has a minimum at $\delta = 0$, so that $h'(0) = 0$. Therefore,

$$\begin{aligned} 0 &= \iiint_\Omega \left(\rho(\mathbf{v} - \mathbf{v}^*) \cdot \mathbf{w} + \frac{d\Psi(F^* + \Delta t \nabla \mathbf{v} + \delta \Delta t \nabla \mathbf{w})}{d\delta} \bigg|_{\delta=0} \right) dV \\ &= \iiint_\Omega \left(\rho(\mathbf{v} - \mathbf{v}^*) \cdot \mathbf{w} + \operatorname{tr} \left(\left(\frac{\partial \Psi}{\partial F}(F^* + \Delta t \nabla \mathbf{v}) \right)^T (\Delta t \nabla \mathbf{w}) \right) \right) dV. \end{aligned} \tag{3.21}$$

We will proceed to analyze the trace integral.

Note that trace is a linear function and the trace of a product is invariant under cyclic permutations. These facts, in conjunction with (3.13), allow us to write

$$\iiint_\Omega \operatorname{tr} \left(\left(\frac{\partial \Psi}{\partial F}(F^* + \Delta t \nabla \mathbf{v}) \right)^T (\Delta t \nabla \mathbf{w}) \right) dV = \Delta t \iiint_\Omega \operatorname{tr} \left(\nabla \mathbf{w} P^T \right) dV, \tag{3.22}$$

where we suppress the fact that the first Piola-Kirchoff stress tensor is evaluated at $F^* + \Delta t \nabla \mathbf{v}$. Write $\mathbf{w} = (w_1, w_2, w_3)^T$, and let $\mathbf{P}_{(\mathbf{i})}$ denote the i-th row of P. Then integration by parts yields

$$\begin{aligned} \iiint_\Omega \operatorname{tr} \left(\nabla \mathbf{w} P^T \right) dV &= \sum_{i=1}^3 \iiint_\Omega \nabla w_i \cdot \mathbf{P}_{(\mathbf{i})} dV \\ &= \sum_{i=1}^3 \left(\iint_{\partial\Omega} w_i \mathbf{P}_{(\mathbf{i})} d\mathbf{N} - \iiint_\Omega w_i \nabla \cdot \mathbf{P}_{(\mathbf{i})} dV \right) \\ &= \iint_{\partial\Omega} \mathbf{w}^T P d\mathbf{N} - \iiint_\Omega \left(\nabla \cdot P^T \right) \cdot \mathbf{w} dV. \end{aligned} \tag{3.23}$$

Substituting this expression back into (3.21) yields

$$\iiint_\Omega \left(\rho(\mathbf{v} - \mathbf{v}^*) - \Delta t \nabla \cdot P^T \right) \cdot \mathbf{w} dV - \Delta t \iint_{\partial\Omega} \mathbf{w}^T P d\mathbf{N} = 0. \tag{3.24}$$

Note that this equality must hold for all $\mathbf{w}$; therefore, we may conclude that

$$\rho(\mathbf{v} - \mathbf{v}^*) - \Delta t \nabla \cdot P^T = 0 \tag{3.25}$$

holds everywhere in Ω, or, equivalently,

$$\mathbf{v} = \mathbf{v}^* + \frac{\Delta t}{\rho} \nabla \cdot P^T (F^* + \Delta t \nabla \mathbf{v}). \tag{3.26}$$

That is, by minimizing $\mathcal{F}$, we obtain a velocity field that results from applying a single step of backward Euler (in time, of size Δt) to the dynamics equation (3.18) (recall that effects of gravity are already accounted for in $\mathbf{v}^*$). Additionally, we must have $P\mathbf{N} = \mathbf{0}$ on $\partial\Omega$ from (3.24), so that there is no traction on the boundary; this agrees with Newton's third law of motion, since there is nothing beyond the solid's boundary to supply a reactionary force if $P\mathbf{N} \neq \mathbf{0}$. Ergo, our claim is proved.

We note that Gast and Schroeder [25] proposed use of the optimization form of backward Euler in the discrete setting, and this technique is commonly used in animation.

3.5 Spatial Discretization

Besides elegantly dealing with time stepping, the presented variational point of view also simplifies spatial discretization of the dynamics equation. In particular, if we sample only a finite set of points within the domain for velocity, we find that the objective functional in the minimization problem (3.19) can be expressed as a quadratic. Setting its gradient to zero then yields a linear system of equations that can be efficiently solved by a standard scheme. In this section, we discuss the details of this strategy.

Before delving into spatial discretization, we first describe our notation. Following a finite element methodology, we begin by dividing the solid in its rest configuration into tetrahedra. The vertices in this mesh are called nodes, whereas the tetrahedra are referred to as elements. We let N and T be the set of all nodes and elements in the mesh, respectively, and, for a tetrahedron $t \in T$, we use the notation $i \in t$ to enumerate its nodes. The velocity field of the solid is sampled at the nodes, and can be used to evolve the mesh in time. The position and velocity of node $i \in N$ are represented by $\mathbf{x}_i$ and $\mathbf{v}_i$, respectively. Lastly, the remainder of this chapter uses $\mathbf{x}$ and $\mathbf{v}$ to denote the stacked vectors of nodal positions and velocities, respectively.

3.5.1 Mass Matrix

The objective functional from (3.19) shall be discretized in two parts, corresponding to the summands of the integrand. The first portion, which we call kinetic, will be discussed in the present section; the potential part is discretized in Section 3.5.3.

Towards discretizing the kinetic integral, we distribute the solid's mass among its nodes. This is done by equally distributing the mass of each tetrahedron among its four nodes.

For a node $i \in N$, the assigned mass is denoted by m_i. Then the kinetic integral can be approximated by

$$\sum_{i \in N} \frac{m_i}{2}(\mathbf{v}_i - \mathbf{v}_i^*) \cdot (\mathbf{v}_i - \mathbf{v}_i^*). \tag{3.27}$$

Let M be the diagonal matrix consisting of the masses m_i, each repeated thrice. Then the above expression can be written as

$$\frac{1}{2}(\mathbf{v} - \mathbf{v}^*)^T M(\mathbf{v} - \mathbf{v}^*). \tag{3.28}$$

M can be used to compute the kinetic energy of the object with respect to the discretization, and, as such, is known as the mass matrix. More specifically, it is referred to as the lumped mass matrix, because it is diagonal. Although one can obtain a better approximation of the kinetic integral by interpolating velocities within each element, this strategy yields a denser mass matrix. As we shall describe in Chapter 5, our solid-fluid coupling scheme relies on inversion of the solid mass matrix; this foresight motivates lumping of the masses for each node.

3.5.2 Deformation Gradient

In the same vein as Teran et al. [61], we make the assumption that the deformation map is linear within each element (and piecewise linear throughout the entire domain). Thus, inside each tetrahedron t, the deformation map is given by

$$\phi(\mathbf{X}) = F_t\mathbf{X} + \mathbf{b}_t, \tag{3.29}$$

so that the deformation gradient in t is simply F_t. Suppose that the nodes of t are at $\mathbf{Y}_0, \mathbf{Y}_1, \mathbf{Y}_2, \mathbf{Y}_3$ in the rest state; we let $\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3$ denote the locations of the corresponding nodes in the current configuration. For $i \in \{1, 2, 3\}$, we define the undeformed and deformed edge vectors:

$$\mathbf{d}_i^m = \mathbf{Y}_i - \mathbf{Y}_0, \tag{3.30}$$

$$\mathbf{d}_i^s = \mathbf{y}_i - \mathbf{y}_0. \tag{3.31}$$

It follows that

$$\mathbf{d}_i^s = \phi(\mathbf{Y}_i) - \phi(\mathbf{Y}_0) = F_t(\mathbf{Y}_i - \mathbf{Y}_0) = F_t\mathbf{d}_i^m. \tag{3.32}$$

If we define

$$D_m = \begin{pmatrix} \mathbf{d}_1^m & \mathbf{d}_2^m & \mathbf{d}_3^m \end{pmatrix}, \tag{3.33}$$

$$D_s = \begin{pmatrix} \mathbf{d}_1^s & \mathbf{d}_2^s & \mathbf{d}_3^s \end{pmatrix}, \tag{3.34}$$

then we must have $D_s = F_t D_m$. Assuming that tetrahedron t is initially not degenerate, D_m must be invertible, so that $F_t = D_s D_m^{-1}$. Note that the edge matrix D_m depends only on the reference configuration; thus, its inverse can be precomputed, allowing us to rapidly acquire the deformation gradient in the element simply by forming the edge matrix for the current configuration.

3.5.3 Stiffness Matrix

We are now ready to discretize the potential part of the continuous variational objective function. Note that the integrand must be constant within each element, so that the integral is equal to

$$\sum_{t \in T} V_t \Psi(F_t), \tag{3.35}$$

where V_t is volume of t (in the reference configuration), and F_t is the deformation gradient after the time step. With the help of the approximation scheme presented in Section 3.5.2, the total amount of stored elastic potential energy for a provided configuration is not difficult to calculate.

Recounting our discrete optimization strategy, however, we are interested in differentiating (3.35). To this end, consider the stored energy in a single tetrahedron with vertex locations $\mathbf{y}_0, \mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3$. Stacking these vectors into $\mathbf{y}$, the element's potential energy can be expressed as

$$E = V \Psi(\epsilon(F(D_s(\mathbf{y})))). \tag{3.36}$$

Vectorizing all matrix intermediate functions column-wise, it is straightforward to differentiate the energy using the chain rule. D_s would then be considered a mapping from $\mathbb{R}^{12}$ to $\mathbb{R}^9$:

$$D_s(\mathbf{y}) = \begin{pmatrix} \mathbf{y}_1 - \mathbf{y}_0 \\ \mathbf{y}_2 - \mathbf{y}_0 \\ \mathbf{y}_3 - \mathbf{y}_0 \end{pmatrix} = \begin{pmatrix} -I & I & 0 & 0 \\ -I & 0 & I & 0 \\ -I & 0 & 0 & I \end{pmatrix} \mathbf{y} := A_D \mathbf{y}, \tag{3.37}$$

where I is the 3×3 identity matrix. Assuming that the elements of the rest edge matrix (D_m^{-1}) for this tetrahedron are d_{ij}, we can view write the expression for $F : \mathbb{R}^9 \to \mathbb{R}^9$:

$$F(\mathbf{D_s}) = A_F \mathbf{D_s}, \tag{3.38}$$

where

$$A_F = \begin{pmatrix} d_{11} I & d_{21} I & d_{31} I \\ d_{12} I & d_{22} I & d_{32} I \\ d_{13} I & d_{23} I & d_{33} I \end{pmatrix}. \tag{3.39}$$

Note that the linearized strain tensor is symmetric; thus, ϵ can be viewed as a function between $\mathbb{R}^9$ and $\mathbb{R}^6$. In this way, we can write

$$\epsilon(\mathbf{F}) = A_\epsilon \mathbf{F} - \mathbf{w}, \tag{3.40}$$

with

$$A_\epsilon = \frac{1}{2}\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}, \quad \mathbf{w} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}. \tag{3.41}$$

Lastly, the function $\Psi : \mathbb{R}^6 \to \mathbb{R}$ is

$$\Psi\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{pmatrix} = \mu(a_1^2 + a_2^2 + a_3^2 + 2a_4^2 + 2a_5^2 + 2a_6^2) + \frac{\lambda}{2}(a_1 + a_2 + a_3)^2, \tag{3.42}$$

so that

$$\begin{aligned} \frac{\partial \Psi}{\partial \mathbf{a}} &= \begin{pmatrix} 2\mu a_1 + \lambda(a_1 + a_2 + a_3) \\ 2\mu a_2 + \lambda(a_1 + a_2 + a_3) \\ 2\mu a_3 + \lambda(a_1 + a_2 + a_3) \\ 4\mu a_4 \\ 4\mu a_5 \\ 4\mu a_6 \end{pmatrix}^T \\ &= \mathbf{a}^T \begin{pmatrix} 2\mu + \lambda & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & 2\mu + \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & 2\mu + \lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & 4\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 4\mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 4\mu \end{pmatrix} := \mathbf{a}^T A_\Psi. \end{aligned} \tag{3.43}$$

With these observations, the potential energy function for the tetrahedron can be differentiated:

$$\begin{aligned} \frac{\partial E}{\partial \mathbf{y}} &= V\epsilon(F(D_s(\mathbf{y})))^T A_\Psi A_\epsilon A_F A_D = V(A_\epsilon A_F A_D \mathbf{y} - \mathbf{w})^T A_\Psi A_\epsilon A_F A_D \\ &= V\mathbf{y}^T A_D^T A_F^T A_\epsilon^T A_\Psi A_\epsilon A_F A_D - V\mathbf{w}^T A_\Psi A_\epsilon A_F A_D := \mathbf{y}^T K_t - \mathbf{w} L_t. \end{aligned} \tag{3.44}$$

The matrix K_t is called the local stiffness matrix for the element t. If we look at potential energy as a function of all nodal positions, then the total energy for the entire solid can be differentiated to obtain the derivative:

$$\sum_{t \in T} (\mathbf{x}^T K_t - \mathbf{w}^T L_t) := \mathbf{x}^T K - \mathbf{b}^T. \tag{3.45}$$

K is known as the solid's stiffness matrix. Note that the matrix A_Ψ is positive-definite, since it is symmetric and diagonally dominant. Hence, we can write it as $B^T B$ for some B; in turn, this allows us to express each local stiffness matrix in the form $C^T C$, where $C = BA_\epsilon A_F A_D$. Since it is a sum of positive-semidefinite matrices, the global stiffness matrix must itself be positive-semidefinite. This important fact allows for application of specialized linear solvers for numerical integration, and will be exploited for efficient positive-definite transformation of the coupled solid and fluid system.

3.5.4 Discrete Equations

Discretized versions of the kinetic and potential parts of the continuous objective function can be added to obtain an optimization-based integrator for the solid. Thanks to the simplicity of the objective function, we can solve the optimization problem simply by setting the gradient to zero. Differentiating the discrete objective function yields

$$(\mathbf{v} - \mathbf{v}^*)^T M + \Delta t (\mathbf{x}^* + \Delta t \mathbf{v})^T K - \mathbf{b}^T. \tag{3.46}$$

We transpose this formula and equate it to zero, obtaining

$$M(\mathbf{v} - \mathbf{v}^*) + \Delta t (K(\mathbf{x}^* + \Delta t \mathbf{v}) - \mathbf{b}) = \mathbf{0}, \tag{3.47}$$

or, equivalently,

$$\left(\frac{1}{\Delta t} M + \Delta t K \right) \mathbf{v} = \frac{1}{\Delta t} M \mathbf{v}^* - (K \mathbf{x}^* - \mathbf{b}). \tag{3.48}$$

Let $\mathbf{x}_0$ be the nodal coordinates in the rest state; consider applying the above equation to a stationary solid in the rest state, in the absence of external forces. Then $\mathbf{v}^*$ and $\mathbf{v}$ must both be zero, implying $\mathbf{b} = K\mathbf{x}_0$. In addition, we remark that external forces can simply be added to the right hand side of the expression. Thus, the discrete equations of motion for the solid are

$$\left(\frac{1}{\Delta t} M + \Delta t K \right) \mathbf{v} = \frac{1}{\Delta t} M \mathbf{v}^* - K(\mathbf{x}^* - \mathbf{x}_0) + \mathbf{f}, \tag{3.49}$$

where $\mathbf{f}$ is the stacked vector of externally-applied nodal forces.

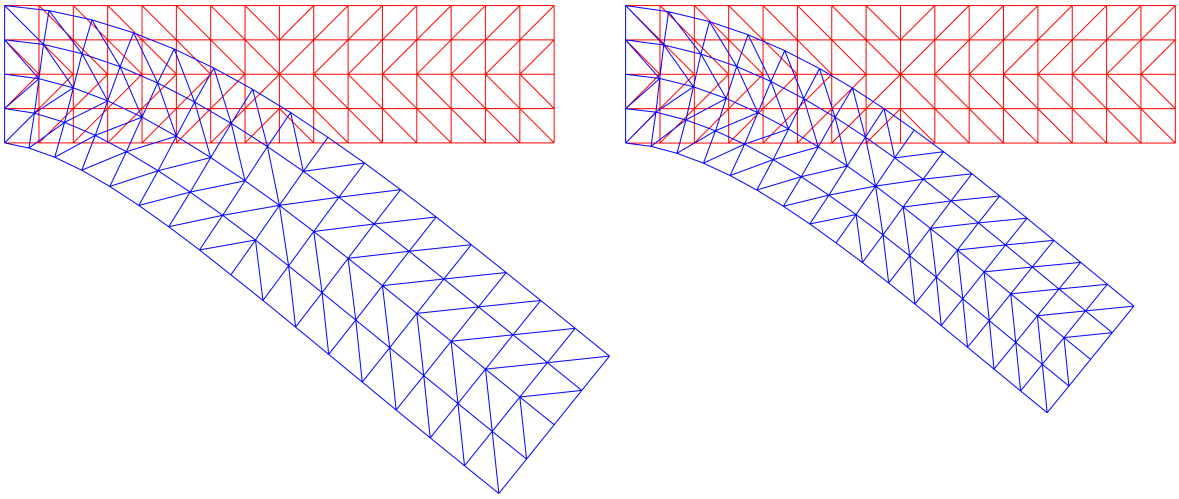

Figure 3.4: Comparison of pure linear (left) and corotational (right) models. Notice how the former suffers from unrealistic volume growth in rotated triangles.

3.5.5 Rayleigh Damping

The presented hyperelastic material model is fully conservative; to introduce energy dissipation to this system, we use the Rayleigh model. In mathematical terms, the damping matrix for this linear damping model is defined as

$$D = \tau_M M + \tau_K K, \tag{3.50}$$

where τ_M, τ_K are non-negative constants. The former parameter controls the degree to which the object's absolute velocity decays, whereas the latter constant governs the amount of damping on the stiffness modes of relative velocities. With this damping model incorporated, the solid equations of motion become

$$\left(\frac{1}{\Delta t}M + D + \Delta t K\right)\mathbf{v} = \frac{1}{\Delta t}M\mathbf{v}^* - K(\mathbf{x}^* - \mathbf{x}_0) + \mathbf{f}. \tag{3.51}$$

3.6 Corotational Model

Recall our motivation for using the described linearization of Green strain: its implicit integration yields a linear system of equations. Yet the resultant model suffers from a

serious deficiency: it is not rotationally invariant. As a result, simulations using this method often exhibit strange artefacts; in particular, non-physical volume growth due to rotations is a recurring issue (see Figure 3.4: the beam, whose initial configuration is given in red, is pinned on the left and allowed to fall under gravity). Fortunately, it is not very difficult to fix this problem, by compensating for rotations on a per-element basis according to the corotational model described by Müller and Gross [42].

For the purpose of explaining the overall idea of this model, we consider a single tetrahedron t. Suppose that we know the manner by which this element has rotated from the rest state; let R_t denote its rotation matrix. As argued in Section 3.5.3, elastic forces experienced by the four nodes of this tetrahedron are given by $\mathbf{f}_t = K_t(\mathbf{x} - \mathbf{x}_0)$. The effect of elemental rotation on the dynamics can be nullified by applying the inverse transformation to the current configuration of the tetrahedron and re-rotating the resultant internal forces. Mathematically,

$$\mathbf{f} = \tilde{R}_t K_t(\tilde{R}_t^T \mathbf{x} - \mathbf{x}_0) = \tilde{R}_t K_t \tilde{R}_t^T \mathbf{x} - \tilde{R}_t K_t \mathbf{x}_0, \tag{3.52}$$

where $\tilde{R}_t$ is a block diagonal matrix formed by repetition of R_t. The overall effect of this corotational model on the described system is augmentation of the global stiffness matrix with the elemental rotations; if we define

$$K = \sum_{t \in T} \tilde{R}_t K_t \tilde{R}_t^T, \tag{3.53}$$

$$\mathbf{b} = \sum_{t \in T} \tilde{R}_t K_t \mathbf{x}_0, \tag{3.54}$$

then the modified equations of motion are

$$\left(\frac{1}{\Delta t} M + D + \Delta t K\right) \mathbf{v} = \frac{1}{\Delta t} M \mathbf{v}^* - K \mathbf{x}^* + \mathbf{b} + \mathbf{f}. \tag{3.55}$$

3.6.1 Elemental Rotations

It remains only to explain how to extract an element's rotation. Observe that the deformation gradient within a tetrahedron captures not only its shape change, but also its rotation. These two parts of the transformation can be decoupled by computing the polar decomposition for the gradient. In particular, we desire to express the deformation gradient as $F = RU$, where R is a rotation matrix, and U is symmetric and positive-semidefinite. The procedure to do so, as presented in by Etzmuß et al. [20], is described below.

First, we note that

$$F^T F = U^T R^T R U = U^T U = U^2. \tag{3.56}$$

Algorithm 1 Solid Initialization Stage

Input:
list of tetrahedra T
rest node locations $\mathbf{x}_0$
Lamé parameters λ, μ

Output:
mass matrix M
lists of $D_{m,t}^{-1}, K_t, \mathbf{b}_t$ for $t \in T$

1: initialize nodal masses to 0
2: **for** each element $t \in T$ **do**
3: compute volume V_t of t
4: distribute mass $\rho V_t/4$ to each of the four nodes of t
5: compute $D_{m,t}^{-1}$ for t (Section 3.5.2)
6: compute K_t for t (Section 3.5.3)
7: compute $\mathbf{b}_t = K_t \mathbf{x}_0$
8: **end for**
9: assemble diagonal mass matrix M from nodal masses

Let $V^T \Lambda V$ be the eigendecomposition of U^2. Then its square root is easy to calculate:

$$U = V^T \sqrt{\Lambda} V, \tag{3.57}$$

where square root of a diagonal matrix can be computed element-wise. Finally, the elemental rotation matrix can be obtained by removing the effect of U:

$$R = FU^{-1} = FV^T \sqrt{\Lambda^{-1}} V, \tag{3.58}$$

where inversion of Λ is trivial, because it is diagonal.

3.7 Algorithm Summary

We conclude this chapter by providing an overview of the simulation procedure. There are two stages to algorithm: initialization and simulation. The first of these steps is used to analyze the solid structure and pre-compute required matrices; pseudocode for this stage is given in Algorithm 1. The simulation stage consists of a loop to repeatedly integrate the system forward in time (see Algorithm 2). Note that gravity forces are integrated

Algorithm 2 Solid Simulation Stage

Input:

initial nodal positions and velocities $(\mathbf{x}_0, \mathbf{v}_0)$
acceleration due to gravity $\mathbf{g}$
time step size Δt
simulation stop time t_f
output of Algorithm 1

Output:

nodal positions and velocities $\mathbf{x}, \mathbf{v}$ at t_f

set $\mathbf{x} = \mathbf{x}_0$, $\mathbf{v} = \mathbf{v}_0$
set $t = 0$

while $t < t_f$ **do**
    set $K = 0$, $\mathbf{b} = \mathbf{0}$
    for each element $t \in T$ **do**
        compute the deformation gradient, F_t, in t (Section 3.5.2)
        compute elemental rotation matrix, R_t, from F_t (Section 3.6.1)
        add $\tilde{R}_t K_t \tilde{R}_t^T$ to K
        add $\tilde{R}_t \mathbf{b}_t$ to $\mathbf{b}$
    end for

    set $\mathbf{v}^* = \mathbf{v} + \Delta t \mathbf{g}$, $\mathbf{x}^* = \mathbf{x}$
    form and solve the dynamics equation (3.55) for $\mathbf{v}$
    set $\mathbf{x} = \mathbf{x}^* + \Delta t \mathbf{v}$
    handle contact with walls
    increment t by Δt
end while

into the velocities before solution of the dynamics equations; as a result, in the absence of other external forces, the force vector in (3.55) is $\mathbf{0}$. This mathematically-equivalent formulation is more consistent with the manner in which fluids are treated, and, as our goal is simulation of interactions between deformables and fluids, it is appropriate that we adopt it for solids as well.

To handle contact with walls, we use a simple explicit correction. After the solid is moved with updated velocities, nodes that have wandered into a wall are identified and projected out. In addition, the normal component of velocity for a penetrating node is

eliminated, simulating an inelastic collision.

Chapter 4

Fluid Dynamics

In many ways, fluids and solids are very much alike. On a smaller scale, both are forms of matter, consisting of the same atoms and molecules. Macroscopically, they can similarly be manipulated through application of forces, and fluids and solids both seem to obey the same laws of conservation of mass, energy, and momentum. Critically, one must allow for arbitrary changes to regions occupied by matter when studying entities of each type. It is fitting, then, that both are modelled as continuum matter, obliging us to to concede that fluid motion is governed by the same set of equations that apply to solids (as presented in Chapter 3).

By the same token, there is a perceptible difference in physical behaviour between solids and fluids. An elastic solid, for instance, will always attempt to revert experienced shape changes, whereas a liquid is happy to accept the form of its enclosure, whatever it may be. As one might expect, this contrasting behaviour is managed through use of different constitutive equations; that is, the conditions that induce stress within a fluid are not the same as a solid's. We will attempt to determine these conditions, and, consequently, the appropriate constitutive laws for the fluids relevant to this thesis: inviscid, incompressible liquids.

Imagine stirring a pot filled with a liquid. The difficulty of this task strongly depends on the type of the liquid: water, for example, is quite easy to stir, while honey requires much more effort. The resistance of fluid matter to being stirred is a consequence of viscosity and is caused by internal shearing forces that arise due to relative motion within the fluid. We use the scenario depicted in Figure 4.2 to illustrate this point. In this diagram, two infinite layers of liquid ($\mathcal{L}_1$ and $\mathcal{L}_2$) are moving in opposite directions. We consider the forces acting on a small square region within the fluid, denoted by Σ, whose four sides are

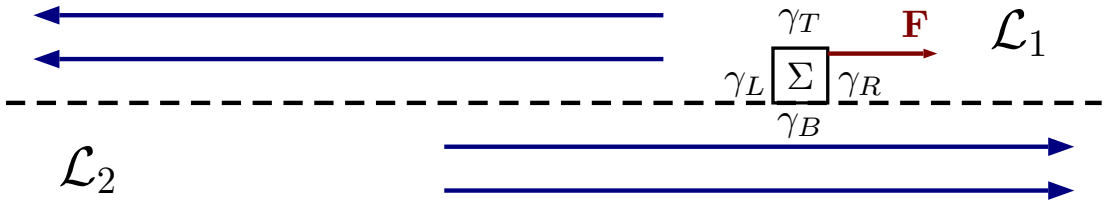

Figure 4.1: Two layers of a fluid moving in opposite directions. Blue arrows indicate velocities, while the red arrow is a force experienced by Σ.

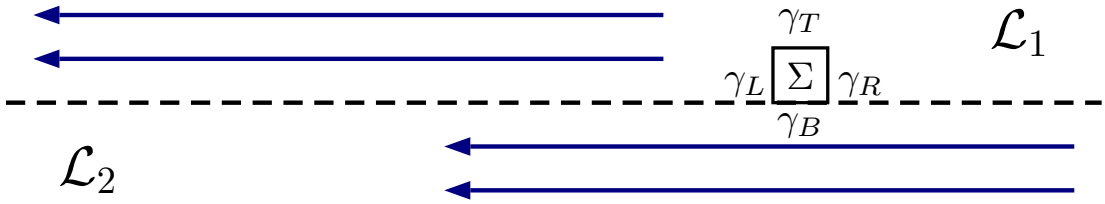

Figure 4.2: Two layers of a fluid moving in the same manner; note that no net force is now generated at Σ.

defined as $\gamma_L, \gamma_R, \gamma_T, \gamma_B$ (note that γ_B is located on the boundary separating $\mathcal{L}_1$ and $\mathcal{L}_2$). If this liquid is viscous, then Σ will experience an internal force $\mathbf{F}$ in the opposite direction of its motion (resisting the relative flow); we are interested in determining the manner in which this force arises.

To contrast, consider the case where both layers are moving in the same direction, as portrayed in Figure 4.2, so that Σ does not experience any forces slowing it down. In this scenario, the conditions within the fluid domain are identical along any vertical cut; in particular, the Cauchy stress tensor, $\tilde{\sigma}$, must be independent of the vertical coordinate. Therefore, since internal force can be obtained by integrating tractions along the boundary, we have

$$\begin{aligned} \mathbf{0} = \int_{\partial\Sigma} \tilde{\sigma} d\mathbf{n} &= \int_{\gamma_L} \tilde{\sigma} d\mathbf{n} + \int_{\gamma_R} \tilde{\sigma} d\mathbf{n} + \int_{\gamma_T} \tilde{\sigma} d\mathbf{n} + \int_{\gamma_B} \tilde{\sigma} d\mathbf{n} \\ &= \int_{\gamma_L} \tilde{\sigma} d\mathbf{n} + \int_{\gamma_R} \tilde{\sigma} d\mathbf{n} + \int_{\gamma_T} \tilde{\sigma} d\mathbf{n} - \int_{\gamma_T} \tilde{\sigma} d\mathbf{n} = \int_{\gamma_L} \tilde{\sigma} d\mathbf{n} + \int_{\gamma_R} \tilde{\sigma} d\mathbf{n}, \end{aligned} \tag{4.1}$$

where we used the fact that the normals along γ_T and γ_B are opposite. Let σ be the Cauchy stress tensor for the non-uniform flow (case in Figure 4.1). Since the stress tensor at a point depends on the state of matter in an infinitesimal neighbourhood around it, we may conclude that $\sigma = \tilde{\sigma}$ on $\gamma_L, \gamma_T, \gamma_B$ (for conditions in $\mathcal{L}_1$ are identical in both scenarios). As such,

$$\mathbf{F} = \int_{\gamma_L} \tilde{\sigma} d\mathbf{n} + \int_{\gamma_R} \tilde{\sigma} d\mathbf{n} + \int_{\gamma_T} \sigma d\mathbf{n} + \int_{\gamma_B} \sigma d\mathbf{n} = \int_{\gamma_T} \sigma d\mathbf{n} + \int_{\gamma_B} \sigma d\mathbf{n} := \mathbf{F}_T + \mathbf{F}_B. \quad (4.2)$$

Denoting the normal to γ_B by $\mathbf{n}$, we see that

$$\mathbf{F}_B^s := \mathbf{F}_B - \mathbf{F}_B \cdot \mathbf{n} \neq \mathbf{0}. \quad (4.3)$$

This force component acts in the tangent plane of the interface and is called the shearing force (see Figure 4.3). Note that, by symmetry, this component is non-existent in the case for uniform flow. This fact is a defining characteristic of fluids: no internal shearing forces arise for a fluid under uniform motion. An inviscid fluid is defined such that shearing forces never arise; that is, tractions are always normal to the surface: $\sigma\mathbf{n} = -p\mathbf{n}$ for some p. It follows that $\sigma = -pI$; p is known as the pressure field, and it plays a critical role in ensuring that our inviscid liquids remain incompressible. In this chapter, we will mathematically describe the nature of the pressure field within a liquid, then proceed with discretization of the equations to obtain a numerical simulation scheme. Once again, the material we are presenting is contained within standard sources, such as the books by Lin et al. [36] and Bridson [9].

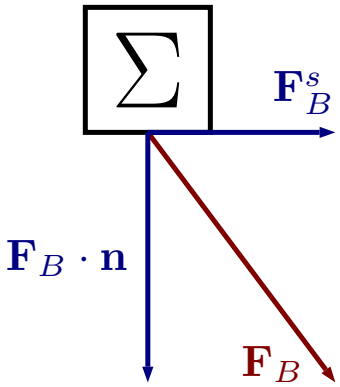

Figure 4.3: Normal ($\mathbf{F}_B \cdot \mathbf{n}$) and shearing ($\mathbf{F}_B^s$) components of traction force.

4.1 Equations of Motion

Aside from necessitating use of different constitutive equations, the unique behaviour of liquids has further repercussions in the way they are normally treated. In contrast to the Lagrangian representation of solids, fluids are generally treated in an Eulerian manner, so that the relevant quantities are measured with respect to a frame of reference fixed in space. Consequently, before introducing the Eulerian equations of motion for fluids,

we must introduce a construct to help us reconcile this viewpoint with the Lagrangian perspective. This construct is called the material derivative and it facilitates prediction of time-evolution of moving quantities, as perceived by a fixed observer.

4.1.1 Material Derivative

Let $\mathbf{X}$ be a material point inside the initial fluid domain. Suppose that this point has some property, say $B(\mathbf{X})$, and assume for simplicity that this property does not change in time. For example, the fluid might be dyed, and B could denote the amount of dye the point carries. We are interested in measuring this quantity with respect to a different frame of reference, one that is not attached to the particle, but rather fixed in space. To this end, consider a fixed location in space, say $\mathbf{x} = (x, y, z)$. Although the property B for a specific particle is constant, its observed value at $\mathbf{x}$ changes with time due to the flow (the dyed points, for instance, could disperse throughout the fluid domain). Let $b(t, \mathbf{x})$ be the measured value at time t and location $\mathbf{x}$; similarly, define $\mathbf{u}(t, \mathbf{x}) = (u_x, u_y, u_z)$ to be the fluid velocity. Then we can use the chain rule to deduce that the total time derivative is

$$\frac{\partial b}{\partial t} + \frac{\partial b}{\partial x}u_x + \frac{\partial b}{\partial y}u_y + \frac{\partial b}{\partial z}u_z = \frac{\partial b}{\partial t} + \mathbf{u} \cdot \nabla b. \tag{4.4}$$

On the other hand, the total derivative is 0, as B is independent of time; thus, we have

$$\frac{\partial b}{\partial t} + \mathbf{u} \cdot \nabla b = 0. \tag{4.5}$$

This expression defines the advection equation; it summarizes time evolution of an Eulerian quantity that is constant in the Lagrangian perspective. The left hand side of the equation is known as the material derivative of b and it is commonly denoted by D/Dt. The property can also be vector-valued ($\mathbf{b} = (b_1, \ldots, b_k)$), in which case its material derivative is given by

$$\frac{D\mathbf{b}}{Dt} = \frac{\partial \mathbf{b}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{b}. \tag{4.6}$$

To be explicit, $\mathbf{u} \cdot \nabla \mathbf{b}$ is a vector, whose i-th element is given by $\mathbf{u} \cdot \nabla b_i$.

4.1.2 Conservation of Mass

The material derivative is an indispensable tool that allows us to track the time-evolution of quantities that are carried along with the flow, from an Eulerian perspective. In the

following sections, we shall utilize it in order to acquire the equations of motion governing inviscid, incompressible fluid flow.

We begin with conservation of mass. Consider a fixed region in space, say Σ; the fluid mass within Σ is given by

$$m = \iiint_{\Sigma} \rho dV, \tag{4.7}$$

where ρ is the Eulerian density field. Since mass cannot be created or destroyed, changes in m must result from fluid entering and leaving the domain. Defining $\mathbf{u}$ to be the Eulerian velocity field, our observation leads us to conclude that

$$\frac{dm}{dt} = \iint_{\partial\Sigma} \rho(-\mathbf{u}) d\mathbf{n} = -\iint_{\partial\Sigma} \rho \mathbf{u} d\mathbf{n} = -\iiint_{\Sigma} \nabla \cdot (\rho \mathbf{u}) dV, \tag{4.8}$$

where we used the divergence theorem for conversion to a volume integral. On the other hand, (4.7) yields

$$\frac{dm}{dt} = \frac{d}{dt} \iiint_{\Sigma} \rho dV = \iiint_{\Sigma} \frac{\partial \rho}{\partial t} dV. \tag{4.9}$$

Since the above equations hold for all Σ, we are forced to conclude that

$$0 = \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = \frac{\partial \rho}{\partial t} + \nabla \rho \cdot \mathbf{u} + \rho \nabla \cdot \mathbf{u} \tag{4.10}$$

holds everywhere. But the fluid is incompressible, implying that the Lagrangian density field must remain constant; that is,

$$0 = \frac{D\rho}{Dt} = \frac{\partial \rho}{\partial t} + \mathbf{u} \cdot \nabla \rho. \tag{4.11}$$

Subtracting equation (4.11) from equation (4.10) gives $\rho \nabla \cdot \mathbf{u} = 0$, or, equivalently, the incompressibility condition:

$$\nabla \cdot \mathbf{u} = 0. \tag{4.12}$$

4.1.3 Conservation of Momentum

We now seek to enforce conservation of momentum in Σ (denoted by p). Note that its time evolution can be summarized by

$$\frac{dp}{dt} = \frac{d}{dt} \iiint_{\Sigma} \rho \mathbf{u} dV = \iiint_{\Sigma} \frac{\partial (\rho \mathbf{u})}{\partial t} dV = \iiint_{\Sigma} \left(\frac{\partial \rho}{\partial t} \mathbf{u} + \rho \frac{\partial \mathbf{u}}{\partial t} \right) dV. \tag{4.13}$$

Change in momentum can be attributed to three mechanisms: transport as a result of flow, effect of internal pressure force, and effect of externally-induced accelerations (such as gravity). We shall discuss each in turn.

To capture the net amount of momentum entering and leaving Σ as a result of flow, we can once again compute the boundary integral:

$$\iint_{\partial\Sigma} \rho\mathbf{u}(-\mathbf{u}\cdot\mathbf{n})dA = -\iint_{\partial\Sigma} \rho\mathbf{u}\mathbf{u}^T d\mathbf{n} = -\iiint_{\Sigma} \nabla\cdot\left(\rho\mathbf{u}\mathbf{u}^T\right) dV. \tag{4.14}$$

Suppose $\mathbf{u} = (u_1, u_2, u_3)$; then the i-th component of the above integral is

$$\begin{aligned} -\iiint_{\Sigma} \nabla\cdot(\rho u_i\mathbf{u})dV &= -\iiint_{\Sigma} \left((\rho\nabla u_i + u_i\nabla\rho)\cdot\mathbf{u} + \rho u_i\nabla\cdot\mathbf{u}\right) dV \\ &= -\iiint_{\Sigma} \left(\rho\mathbf{u}\cdot\nabla u_i + u_i(\mathbf{u}\cdot\nabla\rho + \rho\nabla\cdot\mathbf{u})\right) dV. \end{aligned} \tag{4.15}$$

Therefore, we may express the integral from (4.14) as

$$-\iiint_{\Sigma} \left(\rho\mathbf{u}\cdot\nabla\mathbf{u} + (\mathbf{u}\cdot\nabla\rho)\mathbf{u}\right) dV, \tag{4.16}$$

where we used the fact that $\nabla\cdot\mathbf{u} = 0$.

Since the Cauchy stress tensor for an inviscid fluid is given by $-pI$, the internal pressure force on Σ can be succinctly expressed as

$$\iint_{\partial\Sigma} -pI\,d\mathbf{n} = -\iint_{\partial\Sigma} p d\mathbf{n} = -\iiint_{\Sigma} \nabla p dV. \tag{4.17}$$

Lastly, if $\mathbf{a}$ is the vector of externally-induced accelerations, then the force felt within Σ is

$$\iiint_{\Sigma} \rho\mathbf{a} dV. \tag{4.18}$$

Combining equation (4.13) with equations (4.16)-(4.18) and noting that the equality must be true for all Σ, we deduce that

$$\frac{\partial\rho}{\partial t}\mathbf{u} + \rho\frac{\partial\mathbf{u}}{\partial t} = -\left(\rho\mathbf{u}\cdot\nabla\mathbf{u} + (\mathbf{u}\cdot\nabla\rho)\mathbf{u}\right) - \nabla p + \rho\mathbf{a} \tag{4.19}$$

holds everywhere. Dividing this equation by ρ and rearranging gives the equivalent expression

$$\mathbf{a} = \frac{1}{\rho}\left(\frac{\partial\rho}{\partial t} + \mathbf{u}\cdot\nabla\rho\right)\mathbf{u} + \frac{\partial\mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} + \frac{\nabla p}{\rho} = \frac{\partial\mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} + \frac{\nabla p}{\rho}, \tag{4.20}$$

where we used the fact that the material derivative of the density field is zero (as stated in equation (4.11)) to eliminate the term in parentheses.

4.1.4 Incompressible Euler Equations

The derived equations capture the evolution of incompressible and inviscid fluid flow. They are collectively known as the incompressible Euler equations, and, for completeness, we record them below:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{\nabla p}{\rho} = \mathbf{a}, \tag{4.21}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{4.22}$$

To reiterate, $\mathbf{u}$ is the Eulerian velocity field, p is the pressure field, ρ is fluid density, and $\mathbf{a}$ denotes externally-induced accelerations. Of these two equalities, the first is known as the momentum equation, whereas the second is referred to as the incompressibility condition.

We conclude our discussion of the governing equations by listing the applicable boundary conditions. Let Γ denote the boundary of the fluid region, and partition it into Γ_s (the part in contact with solids) and Γ_a (the free surface). As explained previously, no forces are exerted on the fluid at the free surface; thus, no internal forces can be generated, so that $p\mathbf{n} = \mathbf{0}$, or simply $p = 0$ on Γ_a. On Γ_s, we would like to prevent the liquid from entering the solid, without affecting tangential flow. This condition can be succinctly expressed as $\mathbf{u} \cdot \mathbf{n} = \mathbf{v} \cdot \mathbf{n}$ on Γ_s, where $\mathbf{v}$ was used to denote the solid velocity field.

4.2 Discretization

Numerical solution of the incompressible Euler equations necessitates discrete spatial sampling of the continuous velocity and pressure fields for the liquid. Considering the need for approximating gradient and divergence operators, the simplest discretization scheme might collocate these samples at the nodes of a uniform cubic grid. With such an approach, however, we must either settle for use of first order estimates of derivatives or employ central differencing at a lower resolution. These issues can be circumvented by utilizing the staggered marker-and-cell (MAC) grid, which places pressure samples at cell centres, and velocity components on cell faces. In two dimensions, x components of the velocity field are sampled at centres of vertical faces, while y components sit on horizontal faces (see Figure 4.4). In three dimensions, x components live at the centres of cell faces that are perpendicular to the x axis (the setup is similar for y and z components, as depicted in Figure 4.5). Since its initial introduction to computer graphics by Foster and Metaxas [23], the MAC grid has become the standard discretization scheme for fluid animation, as it allows for straightforward central finite difference approximation of the arising differential operators and avoids odd-even pressure decoupling artefacts of collocated approaches.

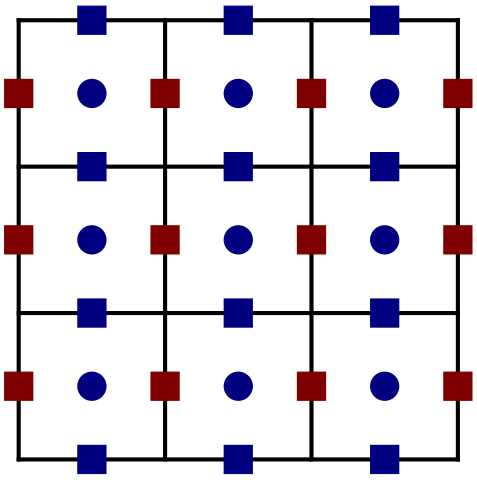

Figure 4.4: A two-dimensional MAC grid; pressure, x and y components of velocity are sampled at blue circles, red squares, and blue squares, respectively.

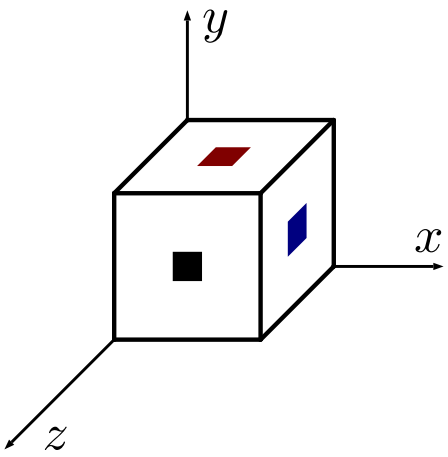

Figure 4.5: Three-dimensional MAC cell; x, y, z components of velocity are sampled at blue, red, black squares, respectively. Pressure sample is at the cell's centre.

Time stepping of the incompressible Euler equations is normally achieved through a technique called operator splitting. This method relies on decoupling the various mechanisms that influence the evolution of liquid velocity, allowing for their effects to be incorporated sequentially. Precisely, a single time step is separated into three stages:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{0}, \tag{4.23}$$

$$\frac{\partial \mathbf{u}}{\partial t} = \mathbf{a}, \tag{4.24}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\nabla p}{\rho} = \mathbf{0} \text{ such that } \nabla \cdot \mathbf{u} = 0. \tag{4.25}$$

The first of these stages is known as advection; it handles the effects of fluid flow on the velocity field. The third stage is referred to as pressure projection and it is responsible for ensuring that the liquid volume does not compress or expand; in addition, the appropriate boundary conditions on the velocity are enforced during pressure projection. We will describe the details of these steps in subsequent sections. Note that the second stage simply adds the effects of external forces and can be solved using a standard numerical integrator; in case of a uniform acceleration (such as gravity), forward Euler is adequate.

4.3 Semi-Lagrangian Advection

Before presenting the numerical scheme that we chose to utilize, we note that our pressure projection algorithm is agnostic to the advection method. Consequently, one is free to solve the advection step using any reasonable technique. As is explained later, we adopt the standard semi-Lagrangian method due to its simplicity and stability.

The advection stage of a simulation step considers the equation

$$\frac{D\mathbf{u}}{Dt} = \mathbf{0}. \tag{4.26}$$

As discussed in Section 4.1.1, this differential equation states that observed changes in $\mathbf{u}$ are only due to the flow. That is, fluid points carry their velocity along, unchanged, as they move around in the domain. Noting this fact, consider advancing the initial velocity field $\mathbf{u}_0(\mathbf{x})$ at time $t = t_0$ to $t = t_0 + \Delta t = t_1$ to obtain $\mathbf{u}_1(\mathbf{x})$. For a fluid point $\mathbf{x}$ in the domain at time t_1, we let $\mathbf{x}_0(\mathbf{x})$ denote its location at time t_0. Then it follows that

$$\mathbf{u}_1(\mathbf{x}) = \mathbf{u}_0(\mathbf{x}_0(\mathbf{x})). \tag{4.27}$$

As such, to sample the new velocity field at a point, we simply need to query the old velocity field for the point's location at the previous time. Thus, we only need to determine an expression for $\mathbf{x}_0$.

To get to its current location of $\mathbf{x}$, the point must have followed the fluid's velocity field. This suggests that we can find the point's original position by following the velocity field backwards for the duration of the time step; hence,

$$\dot{\mathbf{x}}_0 = -\mathbf{u}. \tag{4.28}$$

It is not difficult to numerically integrate this equation explicitly (setting $\mathbf{u} = \mathbf{u}_0$). Typically, the midpoint method is deployed; assuming we are interested in finding $\mathbf{x}_0(\mathbf{x})$, this scheme can be expressed as

$$\mathbf{x}^* = \mathbf{x} - \frac{\Delta t}{2}\mathbf{u}_0(\mathbf{x}), \tag{4.29}$$

$$\mathbf{x}_0(\mathbf{x}) \approx \mathbf{x} - \Delta t \mathbf{u}_0(\mathbf{x}^*). \tag{4.30}$$

Recall that $\mathbf{u}_0$ is stored on a staggered MAC grid. Thus, its evaluation at $\mathbf{x}$ and $\mathbf{x}^*$ must involve some kind of interpolation. In practice, the components of the velocity field are determined separately, using linear interpolation. We demonstrate a simplified version of

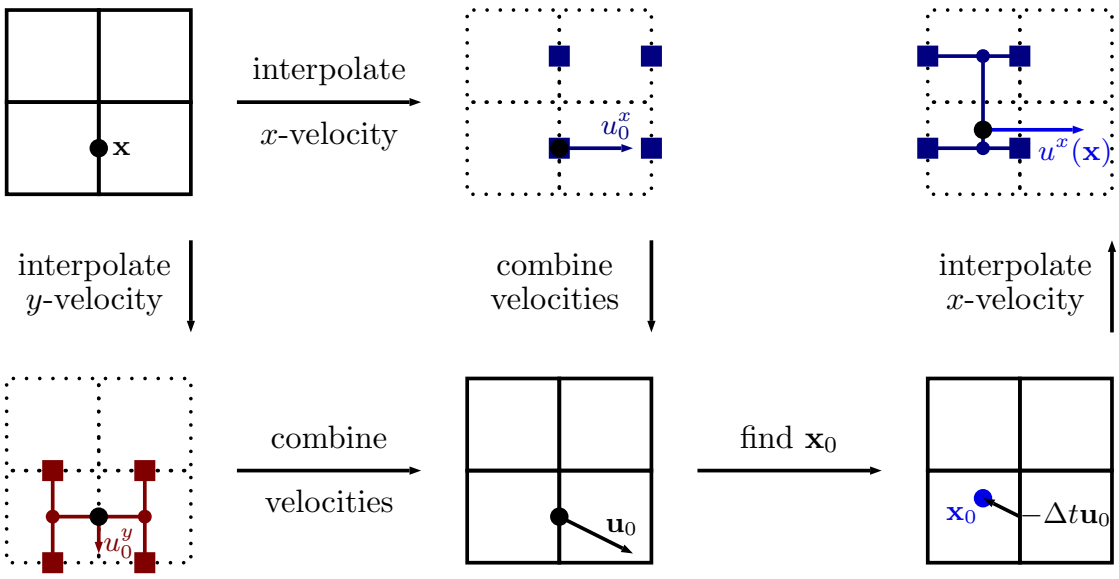

Figure 4.6: Simplified semi-Lagrangian advection on a MAC grid. Objective is to find the x-component of the current velocity field at $\mathbf{x}$ ($u^x(\mathbf{x})$).

this procedure in Figure 4.6 (in two dimensions, using forward Euler instead of explicit midpoint rule).

The described algorithm is aptly termed semi-Lagrangian advection, since it uses insight on behaviour of the Lagrangian velocity field to advect its Eulerian counterpart. It possesses several desirable properties that make it an attractive solution for graphics. In particular, it is simple to implement and it requires no extra data structures beyond the MAC grid. Additionally, it is also incredibly efficient. Crucially, use of linear interpolation renders the scheme unconditionally stable, ensuring that simulations do not blow up as a consequence of taking large time steps.

4.4 Cut-Cell Pressure Projection

An integral part of a liquid simulator, the pressure projection step is responsible for simultaneously ensuring incompressibility and enforcing boundary conditions. In this section, we describe a formulation for this crucial stage of the simulation process, in the presence of solid boundaries (where the free-slip condition applies: $\mathbf{u}\cdot\mathbf{n} = \mathbf{v}\cdot\mathbf{n}$) and the free surface

(where the pressure field must vanish: $p = 0$). For the current chapter, we aim to only capture one-way coupling to static solids (so that $\mathbf{v} = \mathbf{0}$), which is sufficient for simulation of interactions between a liquid and its enclosure. Two-way coupling between liquids and deformable bodies will be discussed in Chapter 5.

4.4.1 Enforcing Incompressibility

The pressure projection stage of a simulation step generally involves finding a pressure field whose effect on the fluid velocity renders it divergence-free. The specific algorithm we shall present in this section, due to Ng et al. [43], uses cut-cells to capture solid geometry, allowing for accurate enforcement of the free-slip condition. This formulation begins by discretizing equation (4.25) in time, yielding

$$\mathbf{u} = \mathbf{u}^* - \frac{\Delta t}{\rho}\nabla p. \tag{4.31}$$

Here, $\mathbf{u}^*$ denotes the velocity field before pressure projection and $\mathbf{u}$ is the velocity field after effects of pressure are factored in, so that $\nabla \cdot \mathbf{u} = 0$.

Consider the fluid region, denoted by Ω, contained within a cubic computational cell (refer to Figure 4.7 for a two-dimensional example); incompressibility in conjunction with the above velocity update rule imply

$$\begin{aligned} 0 &= \iiint_{\Omega} \nabla \cdot \mathbf{u} dV = \iint_{\partial\Omega} \mathbf{u} d\mathbf{n} \\ &= \iint_{\partial\Omega_F} \left(\mathbf{u}^* - \frac{\Delta t}{\rho}\nabla p\right) d\mathbf{n} + \iint_{\partial\Omega_S} \mathbf{v} d\mathbf{n} \\ &= \iint_{\partial\Omega_F} \left(\mathbf{u}^* - \frac{\Delta t}{\rho}\nabla p\right) d\mathbf{n}, \end{aligned} \tag{4.32}$$

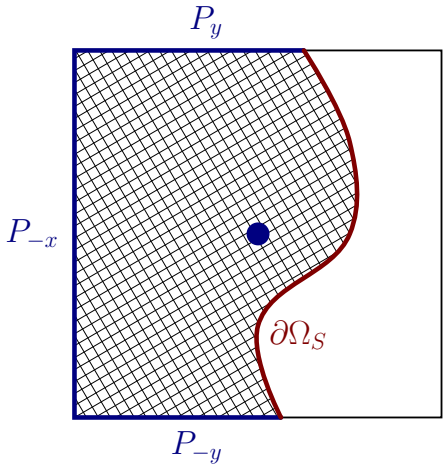

Figure 4.7: A sample configuration within a cell. Ω is shaded, $\partial\Omega_F = P_{-x} \cup P_{-y} \cup P_y$ is in blue.

where the divergence theorem was used to convert the expression into a surface integral. In the last step, we partitioned the liquid's boundary into the regions in contact with the solid, $\partial\Omega_S$, and the remainder, $\partial\Omega_F$, and used the fact that $\mathbf{u}\cdot\mathbf{n} = \mathbf{v}\cdot\mathbf{n} = \mathbf{0}$ on $\partial\Omega_S$. To simplify the derivation, we ignore the presence of a free surface (this boundary

condition is handled separately, as we shall describe in Section 4.4.2). It follows that the non-solid portion of the boundary is a union of axis-aligned planar regions, which are parts of the cell walls neighbouring other fluid cells: $\partial\Omega_F = P_x \cup P_{-x} \cup P_y \cup P_{-y} \cup P_z \cup P_{-z}$, where subscripts are used to denote outward-pointing normals. Consequently, the flux across $\partial\Omega_F$ can be approximated as follows:

$$\iint_{\partial\Omega_F} \mathbf{u}^* d\mathbf{n} \approx \sum_{\alpha\in\{x,y,z\}} \left(A(P_\alpha)u^*(P_\alpha) - A(P_{-\alpha})u^*(P_{-\alpha})\right). \tag{4.33}$$

In the above equation, $A(P)$ is the area of region P (which can be pre-computed for static boundaries), and $u^*(P)$ is the normal component of the intermediate fluid velocity at the center of the face that contains P; by construction, this velocity value can be read directly from the MAC grid. Likewise, we have

$$\iint_{\partial\Omega_F} \nabla p d\mathbf{n} \approx \sum_{\alpha\in\{x,y,z\}} \left(A(P_\alpha)\frac{\partial p}{\partial\alpha}(P_\alpha) - A(P_{-\alpha})\frac{\partial p}{\partial\alpha}(P_{-\alpha})\right), \tag{4.34}$$

where the partial derivatives at cell face centers can be estimated with finite differences between the cell-centered pressures. For example, using subscripts to index into the MAC grid and assuming that cell (i, j, k) is considered, we have

$$\frac{\partial p}{\partial x}(P_x) \approx \frac{p_{i+1,j,k} - p_{i,j,k}}{\Delta x}, \tag{4.35}$$

$$\frac{\partial p}{\partial y}(P_{-y}) \approx \frac{p_{i,j,k} - p_{i,j-1,k}}{\Delta x}, \tag{4.36}$$

$$\frac{\partial p}{\partial z}(P_{-z}) \approx \frac{p_{i,j,k} - p_{i,j,k-1}}{\Delta x} \tag{4.37}$$

and similarly for the other derivatives.

Substituting estimates (4.33) and (4.34) into (4.32) gives the incompressibility equation for a single cell:

$$\begin{aligned} -\frac{\Delta t}{\rho} \sum_{\alpha\in\{x,y,z\}} &\left(A(P_\alpha)\frac{\partial p}{\partial\alpha}(P_\alpha) - A(P_{-\alpha})\frac{\partial p}{\partial\alpha}(P_{-\alpha})\right) \\ &= -\sum_{\alpha\in\{x,y,z\}} \left(A(P_\alpha)u^*(P_\alpha) - A(P_{-\alpha})u^*(P_{-\alpha})\right). \end{aligned} \tag{4.38}$$

We now use $\mathbf{u}^*$ and $\mathbf{p}$ to denote the stacked vector of all discretely sampled intermediate velocities and pressures, respectively. Then, combining the incompressibility condition for

all cells in the domain, we can write the pressure projection step in matrix form

$$B\mathbf{p} = E\mathbf{u}^*. \tag{4.39}$$

In effect, the matrices B and E act as discrete versions of the continuous Laplacian and divergence operators, respectively. This equation can be solved to obtain the pressure field within the liquid; these pressures can then be used to update the fluid velocity field per formula (4.31) (with discretized derivatives). Note that the matrix B above is positive-semidefinite; in fact, it is also non-singular if the redundant cells (those without any liquid) are removed from the system and there is a free surface. As was the case with solid equations, this fact enables application of specialized linear solvers for efficient solution of the system (such as the preconditioned conjugate gradient method).

4.4.2 Free Surface Condition

We have yet to discuss how to satisfy the other applicable boundary condition: at the liquid's free surface, its pressure must drop to 0. Enright et al. [17] outline a simple, yet effective, technique to handle this requirement, called the ghost fluid method. Consider the scenario depicted in Figure 4.8 (where p_1 is located within the fluid, but p_2 is on the "air" side); the aforementioned method notes that the pressure at the exact surface (drawn as a blue dot) must be zero. Thus, the partial derivative at $\mathbf{x}$ can be approximated by

$$\frac{\partial p}{\partial y}(\mathbf{x}) \approx \frac{0 - p_1}{\theta \Delta x} = -\frac{p_1}{\theta \Delta x}. \tag{4.40}$$

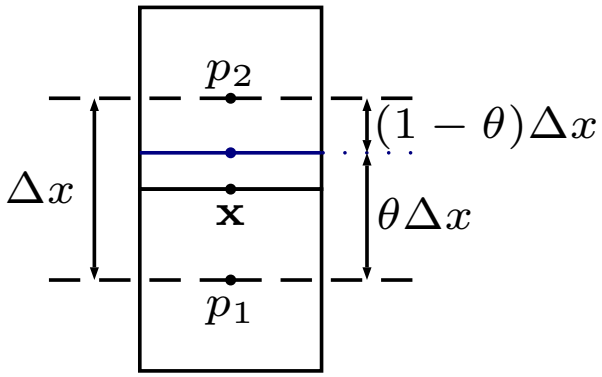

Figure 4.8: A sample configuration for the free surface, depicted by the blue line.

As such, we only need to determine the fraction θ. To this end, the liquid's signed distance function will be utilized.

Take a closed surface $S \subset \mathbb{R}^3$; the signed distance field of S is a function $f : \mathbb{R}^3 \to \mathbb{R}$

such that

$$|f(\mathbf{x})| = \min_{\mathbf{y} \in S} \|\mathbf{y} - \mathbf{x}\|, \tag{4.41}$$

$$\operatorname{sign}(f(\mathbf{x})) = \begin{cases} -1 & \text{if } \mathbf{x} \text{ is in the interior of } S, \\ 1 & \text{otherwise.} \end{cases} \tag{4.42}$$

Intuitively, the signed distance field returns the distance of a point from the surface (with its magnitude) and indicates whether or not the point is in the interior (with its sign). In addition, this function characterizes the set itself, which is simply the locus of points satisfying $f(\mathbf{x}) = 0$. This observation allows us to find the aforementioned θ.

Suppose that values of the fluid's signed distance field are known at locations of samples p_1, p_2; denote these values by ϕ_1, ϕ_2. On the line connecting these two points, linear interpolation can be used to approximate the signed distance field. Particularly, we are interested in finding θ such that $(1 - \theta)\phi_1 + \theta\phi_2 = 0$; solving this equation gives

$$\theta = \frac{\phi_1}{\phi_1 - \phi_2}. \tag{4.43}$$

Finally, this expression can be plugged into (4.40) to obtain the approximation

$$\frac{\partial p}{\partial y}(\mathbf{x}) \approx -\frac{p_1}{\Delta x}\frac{\phi_1 - \phi_2}{\phi_1} = \frac{p_1}{\Delta x}\frac{\phi_2 - \phi_1}{\phi_1}. \tag{4.44}$$

Finally, this approximation of the derivative can be used wherever the free surface is present (in equations (4.31) and (4.38)) to enforce the required boundary condition.

4.5 Tracking Geometry

Our treatment of fluids so far has been exclusively at the velocity level. Since this thesis focuses on simulation for the purpose of animation, the actual liquid body must likewise be tracked. To this end, we fill the initial liquid domain with marker particles and, after each time step of the incompressible Euler equations, advect these particles along the divergence-free velocity field. The velocity of a particle is determined through linear interpolation on the MAC grid, and the particle locations are integrated using the explicit midpoint method, akin to the procedure described in Section 4.3.

Besides facilitating visualization, the marker particles are also used to construct the liquid's signed distance field. This function is sampled at cell centres (as required by the

Algorithm 3 Fluid Simulation Loop

Input:

fluid density
MAC grid solid face areas
list of marker particles
initial (divergence-free) MAC grid velocities
acceleration due to gravity $\mathbf{g} = (g_x, g_y, g_z)$
time step size Δt
simulation stop time t_f

Output:

MAC grid velocities at t_f
list of marker particles at t_f

set $t = 0$
while $t < t_f$ **do**
    advect marker particles (Section 4.5)
    advect MAC grid velocities (Section 4.3)
    add $\Delta t g_x$ to all entries of the x MAC grid
    add $\Delta t g_y$ to all entries of the y MAC grid
    add $\Delta t g_z$ to all entries of the z MAC grid
    compute fluid's signed distance field (Section 4.5)
    perform pressure projection (Section 4.4)
    extrapolate fluid velocities
    increment t by Δt
end while

ghost fluid scheme), using a simple union-of-spheres approximation. That is, each particle is assigned a radius r, so that every point within a distance of r of the said particle is considered to be inside the liquid domain. In other words, the liquid region is determined to be the union of spheres centred at the marker particles, each with radius r; this convention is used for simulation, as well as rendering.

4.6 Algorithm Summary

The overall fluid simulation process is outlined in Algorithm 3. As the steps of this procedure have been described in theoretical detail within previous parts of this chapter, we

presently only make practical implementation comments. We mentioned in passing that the redundant pressure degrees of freedom are a cause for singularity. Thus, eliminating these variables effectively reduces the burden of the deployed numerical solver. Consequently, "active" pressure cells are first identified; a cell is marked active if it contains some fluid and its center is inside either the fluid or solid regions. The pressure projection system is then assembled, involving only these active variables. In addition, face area fractions below 0.01 are clamped down to 0, and fractions above 0.99 are rounded up to 1, to avoid conditioning problems.

Successive advection steps will inevitably query the fluid velocity field outside of the valid region updated by the pressure projection step. We use a simple constrained velocity extrapolation approach to assign fluid grid velocities to these regions [32, 46] (line 10 of the algorithm). That is, for each invalid velocity sample on the MAC grid neighbouring at least one updated entry, we set the value to the average of its valid neighbours. Quantities extrapolated into the solid are then modified to ensure that the normal velocity component is zero. As is the case with semi-Lagrangian advection, to maintain order independence and allow for simple parallelization, the updates are not performed in place, but rather saved in a copy of the array.

Chapter 5

Solid-Fluid Coupling

In the preceding chapters, we presented schemes for simulating elastic solids and fluids in isolation. In this chapter, we describe our novel coupled pressure projection method that simultaneously enforces fluid incompressibility and integrates the solid's internal forces, allowing for accurate satisfaction of boundary conditions at the solid-fluid interface. But before doing so, let us recap the relevant results.

Recall from Chapter 3 that the discrete equations of motion for a deformable body are

$$\left(\frac{1}{\Delta t}M + D + \Delta t K\right)\mathbf{v} = \frac{1}{\Delta t}M\mathbf{v}^* - K\mathbf{x}^* + \mathbf{b} + \mathbf{f}, \tag{5.1}$$

$$\mathbf{x} = \mathbf{x}^* + \Delta t\mathbf{v}. \tag{5.2}$$

In these equations, M, D, K are the solid mass, damping, and stiffness matrices, respectively; $\mathbf{x}$ and $\mathbf{v}$ are the nodal positions and velocities for the body, with asterisks used to identify the corresponding quantities before the time step; lastly, $\mathbf{f}$ is the vector of external forces.

On the liquid side, the pressure projection equation is

$$B\mathbf{p} = E\mathbf{u}^*, \tag{5.3}$$

$$\mathbf{u} = \mathbf{u}^* - \frac{\Delta t}{\rho}G\mathbf{p}. \tag{5.4}$$

where $\mathbf{p}$ is the stacked vector of liquid pressures, $\mathbf{u}$ contains fluid velocities from the MAC grid, and G represents the discrete gradient operator; refer to Chapter 4 for a detailed

discussion on the structure of matrices B and E. Note that every equation in (5.3) corresponds to discretization of

$$\iint_{\partial\Omega_F} \left(\mathbf{u}^* - \frac{\Delta t}{\rho} \nabla p \right) d\mathbf{n} = 0, \tag{5.5}$$

where $\partial\Omega_F$ is the non-solid portion of the cell boundary. Recall that the full equation was

$$\iint_{\partial\Omega_F} \left(\mathbf{u}^* - \frac{\Delta t}{\rho} \nabla p \right) d\mathbf{n} + \iint_{\partial\Omega_S} \mathbf{v} d\mathbf{n} = 0, \tag{5.6}$$

where $\partial\Omega_S$ is the part of $\partial\Omega$ in contact with the solid. As we were dealing with stationary boundaries, this solid integral vanished; in the presence of dynamic bodies, however, the integral must be considered to account for the effect of solid's motion on the liquid.

5.1 Coupled Pressure Projection

We are now ready to describe our coupled pressure projection strategy, which involves fairly straightforward alterations to the restated equations of motion.

5.1.1 Modification to Fluid Equations

The modifications to the fluid incompressibility equations arise from discretization of the solid boundary integral in (5.6). Noting that the deformable's surface is represented by a triangle mesh, we partition $\partial\Omega_S$ into planar polygonal regions $Q_1, \ldots, Q_k$; these regions are not necessarily triangular, since they result from clipping mesh triangles against the cubic fluid cells. Letting a_i and $\mathbf{n}_i$ be the area and unit normal of region Q_i (pointing out of the fluid), and $\mathbf{c}_i$ be the

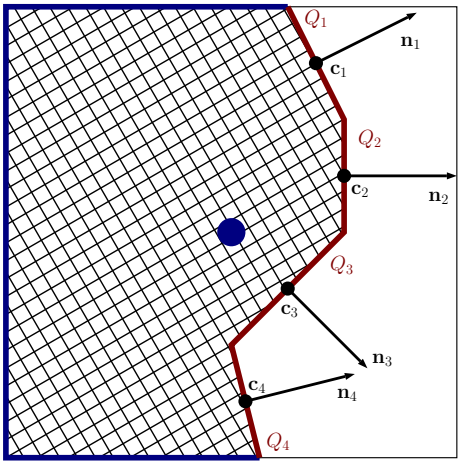

Figure 5.1: A sample configuration within a cell. Ω is shaded, $\partial\Omega_S = Q_1 \cup Q_2 \cup Q_3 \cup Q_4$ is in red.

centroid of Q_i (a simplified two-dimensional scenario is pictured in Figure 5.1), we can use the following approximation:

$$\int_{\partial\Omega_S} \mathbf{v} d\mathbf{n} \approx \sum_{i=1}^{k} a_i \mathbf{v}(\mathbf{c}_i) \cdot \mathbf{n}_i. \tag{5.7}$$

To estimate the velocity at the centroid, $\mathbf{v}(\mathbf{c}_i)$, linear interpolation is used. Specifically, note that Q_i is contained within a surface triangle of the solid, and let the nodes of this triangle be numbered $i(1), i(2), i(3)$. If $\mathbf{x}_j$ is the location of node j, then we can express $\mathbf{c}_i$ in terms of its barycentric coordinates:

$$\mathbf{c}_i = \alpha \mathbf{x}_{i(1)} + \beta \mathbf{x}_{i(2)} + (1 - \alpha - \beta)\mathbf{x}_{i(3)}. \tag{5.8}$$

Finally, this allows us to approximate the velocity at $\mathbf{c}_i$ using

$$\mathbf{v}(\mathbf{c}_i) = \alpha \mathbf{v}_{i(1)} + \beta \mathbf{v}_{i(2)} + (1 - \alpha - \beta)\mathbf{v}_{i(3)}, \tag{5.9}$$

where $\mathbf{v}_j$ is the velocity of node j. Then the sum in (5.7) can be expressed in vector form as $N^T C \mathbf{v}$, where N is the matrix containing area-weighted normals (oriented into the solid), C is the matrix of barycentric coordinates, and $\mathbf{v}$ is the stacked vector of nodal velocities. With this change, the modified pressure equations read

$$-E\mathbf{u}^* + B\mathbf{p} + N^T C \mathbf{v} = \mathbf{0}. \tag{5.10}$$

5.1.2 Modification to Solid Equations

On the solid side, we only need to add the effects of fluid pressure, by incorporating them into the vector of external forces. This is done by observing that the force arising from a pressure field p acting on a surface S is

$$\iint_S p d\mathbf{n}, \tag{5.11}$$

where $\mathbf{n}$ points into the surface.

Consider a planar patch P acquired by intersecting one of the solid's surface triangles T with a fluid cell; we will use p to denote fluid pressure at this cell's center. We make the simplifying assumption that pressure is constant within the computational cell; though this may incur some minor additional error, it is a common and effective choice in graphics

applications [65, 5, 9]). The force applied by the liquid pressure on P is then given by $\mathbf{f} = pA(P)\mathbf{n}$, where $A(P)$ and $\mathbf{n}$ are the area and unit normal of P (oriented into the solid). We further assume that $\mathbf{f}$ is applied at $\mathbf{c}$, the centroid of the patch P, and distribute this force among the nodes of T. Specifically, if the nodes of T are numbered $1, 2, 3$, and the barycentric representation of $\mathbf{c}$ within this triangle is

$$\mathbf{c} = \alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2 + \alpha_3\mathbf{x}_3, \tag{5.12}$$

then the force distributed to node i is

$$\mathbf{f}_i = \alpha_i\mathbf{f} = \alpha_i pA(P)\mathbf{n}. \tag{5.13}$$

Stacking all nodal pressure forces into a vector, we find that it can be expressed as $C^T N\mathbf{p}$, where C and N were defined in the previous section. As such, (5.1) with pressure forces factored in reads

$$\left(\frac{1}{\Delta t}M + D + \Delta t K\right)\mathbf{v} = \frac{1}{\Delta t}M\mathbf{v}^* - K\mathbf{x}^* + \mathbf{b} + C^T N\mathbf{p}. \tag{5.14}$$

5.1.3 Coupled System

Given the solid and fluid discretizations, we can proceed to assemble our combined linear system. We note that (5.10) can be rearranged into

$$B\mathbf{p} + N^T C\mathbf{v} = E\mathbf{u}^*. \tag{5.15}$$

Likewise, (5.14) can be equivalently written as

$$\left(\frac{1}{\Delta t}M + D + \Delta t K\right)\mathbf{v} - C^T N\mathbf{p} = \frac{1}{\Delta t}M\mathbf{v}^* - K\mathbf{x}^* + \mathbf{b}. \tag{5.16}$$

Stacking these last two equations gives the linear system for our coupled pressure projection scheme

$$\begin{pmatrix} B & N^T C \\ C^T N & -\left(\frac{1}{\Delta t}M + D + \Delta t K\right) \end{pmatrix}\begin{pmatrix} \mathbf{p} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} E\mathbf{u}^* \\ -\frac{1}{\Delta t}M\mathbf{v}^* + K\mathbf{x}^* - \mathbf{b} \end{pmatrix} \tag{5.17}$$

which we solve for fluid pressures and solid velocities. This allows us to simultaneously account for fluid incompressibility, solid internal forces, and the two-way interactions between these two entities. Note that, in order to achieve symmetry, we negated the solid equations.

5.2 Positive-Definite Transformation

The matrix associated with the linear system (5.14) for the corotational solid dynamics is positive-definite, a fact that has been exploited in prior work (for example, [31]). While the negation of the solid equations above allows us to obtain a symmetric system, it also causes the principal sub-block corresponding to solid velocities to be negative-definite. Since B is positive-definite, the overall coefficient matrix for our pressure projection solve (5.17) is only symmetric indefinite. Although various methods tailored towards such equations exist (such as MINRES or QMR, as discussed by Robinson-Mosher et al. [49, 47]), it is generally more desirable to work with symmetric positive-definite (SPD) systems. Fortunately, the symmetric coefficient matrix in (5.17) is endowed with a special structure: the principal submatrix corresponding to pressures is positive-definite, and the principal submatrix for solid velocities is negative-definite and constructed as a weighted sum of a stiffness matrix and a diagonal mass matrix. We exploit these facts in order to efficiently convert the system into SPD form; our method to do so is described below.

5.2.1 Algebraic Transformation

We begin by expressing the (negative) principal velocity submatrix as a sum of arbitrary symmetric positive-definite matrices Z_1 and Z_2:

$$\frac{1}{\Delta t}M + D + \Delta t K = Z_1 + Z_2. \tag{5.18}$$

Then the second set of equations in (5.17) reads

$$C^T N\mathbf{p} - Z_1\mathbf{v} - Z_2\mathbf{v} = \mathbf{w}, \tag{5.19}$$

where we let $\mathbf{w}$ denote the right-hand side for convenience. This equation is equivalent to

$$\mathbf{v} = Z_1^{-1}(C^T N\mathbf{p} - Z_2\mathbf{v} - \mathbf{w}). \tag{5.20}$$

Substituting this expression for $\mathbf{v}$ into the first set of equations from (5.17) yields

$$B\mathbf{p} + N^T C Z_1^{-1}(C^T N\mathbf{p} - Z_2\mathbf{v} - \mathbf{w}) = E\mathbf{u}^*, \tag{5.21}$$

which can be rearranged into

$$\left(B + N^T C Z_1^{-1} C^T N\right)\mathbf{p} - N^T C Z_1^{-1} Z_2\mathbf{v} = E\mathbf{u}^* + N^T C Z_1^{-1}\mathbf{w}. \tag{5.22}$$

Also, note that we can use (5.20) to derive

$$\mathbf{0} = Z_2\mathbf{v} - Z_2\mathbf{v} = Z_2\mathbf{v} - Z_2Z_1^{-1}(C^TN\mathbf{p} - Z_2\mathbf{v} - \mathbf{w}), \tag{5.23}$$

which is equivalent to

$$-Z_2Z_1^{-1}C^TN\mathbf{p} + \left(Z_2 + Z_2Z_1^{-1}Z_2\right)\mathbf{v} = -Z_2Z_1^{-1}\mathbf{w}. \tag{5.24}$$

Finally, combining (5.22) and (5.24) gives the following SPD system:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}\begin{pmatrix} \mathbf{p} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} E\mathbf{u}^* + N^TCZ_1^{-1}\mathbf{w} \\ -Z_2Z_1^{-1}\mathbf{w} \end{pmatrix}, \tag{5.25}$$

where

$$\mathbf{w} = -\frac{1}{\Delta t}M\mathbf{v}^* + K\mathbf{x}^* - \mathbf{b}, \tag{5.26}$$

$$A_{11} = B + N^TCZ_1^{-1}C^TN, \tag{5.27}$$

$$A_{12} = -N^TCZ_1^{-1}Z_2, \tag{5.28}$$

$$A_{21} = -Z_2Z_1^{-1}C^TN = A_{12}^T, \tag{5.29}$$

$$A_{22} = Z_2 + Z_2Z_1^{-1}Z_2. \tag{5.30}$$

5.2.2 Choice of Summand Matrices

Note that Z_1^{-1} appears extensively in (5.25); as such, it is ideal to choose Z_1 to be a diagonal matrix. Upon experimenting, we found that letting Z_1 be a multiple of the solid mass matrix provides good results. Considering our use of Rayleigh damping and the constraint that Z_2 be SPD, it is reasonable to set $Z_1 = \sigma(1/\Delta t + \tau_M)M$ (where τ_M is the Rayleigh mass damping coefficient), for $\sigma \in (0, 1)$; we exclusively used $\sigma = 0.9$ to generate simulations in Chapter 6 without encountering numerical difficulties. This construction implies

$$\begin{aligned} Z_2 &= \frac{1}{\Delta t}M + D + \Delta t K - Z_1 \\ &= \left(\frac{1}{\Delta t} + \tau_M\right)M + (\Delta t + \tau_K)K - \sigma\left(\frac{1}{\Delta t} + \tau_M\right)M \\ &= (1-\sigma)\left(\frac{1}{\Delta t} + \tau_M\right)M + (\Delta t + \tau_K)K. \end{aligned} \tag{5.31}$$

5.2.3 Connection to the Schur Complement

Using the cut-cell formalism, the problem of two-way coupling between fluids and rigid bodies can be similarly solved during pressure projection [5, 9], yielding a familiar system

$$\begin{pmatrix} A & J^T \\ J & -\frac{1}{\Delta t}\tilde{M} \end{pmatrix} \begin{pmatrix} \mathbf{p} \\ \mathbf{v} \end{pmatrix} = \mathbf{c}. \tag{5.32}$$

$\tilde{M}$ in the above equation represents the solid mass and inertia matrix, and J captures the effect of pressure forces and torques on the rigid body. As $\tilde{M}$ is only a 6×6 matrix, its inversion does not pose a significant challenge; thus, it is practical to take the Schur complement with respect to $\mathbf{v}$ to get a system in terms of only pressures. Doing so yields the symmetric positive-definite coefficient matrix $A + \Delta t J^T \tilde{M}^{-1} J$ (whereas the full system was symmetric indefinite). We observe that our described transformation technique for deformable bodies is quite similar; in particular, if we duplicate the solid velocities in (5.17) to get

$$\begin{pmatrix} B & N^T C & 0 \\ C^T N & -Z_1 & -Z_2 \\ 0 & -Z_2 & Z_2 \end{pmatrix} \begin{pmatrix} \mathbf{p} \\ \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix} = \begin{pmatrix} E\mathbf{u}^* \\ -\frac{1}{\Delta t} M\mathbf{v}^* + K\mathbf{x}^* - \mathbf{b} \\ \mathbf{0} \end{pmatrix}, \tag{5.33}$$

then our algebraic transformation is equivalent to taking the Schur complement with respect to $\mathbf{v}_1$.

5.2.4 Proof of Positive-Definiteness

It is fairly straightforward to establish that the transformed system (5.25) is indeed SPD. To this end, take vectors $\mathbf{p}, \mathbf{v}$ (at least one of which is non-zero). Because B and Z_2 are both SPD, it can be concluded that

$$\mathbf{p}^T B \mathbf{p} + \mathbf{v}^T Z_2 \mathbf{v} > 0. \tag{5.34}$$

Also, Z_1 is SPD, so its inverse (which is necessarily SPD as well) can be written as $Z_1^{-1} = Y^T Y$ for some matrix Y. Use of these facts allows us to derive

$$\begin{aligned} &\begin{pmatrix} \mathbf{p}^T & \mathbf{v}^T \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} \mathbf{p} \\ \mathbf{v} \end{pmatrix} \\ &\quad = \mathbf{p}^T B \mathbf{p} + \mathbf{p}^T N^T C Z_1^{-1} C^T N \mathbf{p} \\ &\quad - 2\mathbf{p} N^T C Z_1^{-1} Z_2 \mathbf{v} + \mathbf{v}^T Z_2 \mathbf{v} + \mathbf{v}^T Z_2 Z_1^{-1} Z_2 \mathbf{v} \\ &\quad > \mathbf{p}^T N^T C Y^T Y C^T N \mathbf{p} - 2\mathbf{p} N^T C Y^T Y Z_2 \mathbf{v} + \mathbf{v}^T Z_2 Y^T Y Z_2 \mathbf{v} \\ &\quad = \|Y C^T N \mathbf{p} - Y Z_2 \mathbf{v}\|^2 \geq 0, \end{aligned} \tag{5.35}$$

as required.

5.2.5 Matrix Scaling

Lastly, we note that the system (5.25) may suffer from poor conditioning due to different element magnitudes in the solid and fluid blocks. We resolved this issue through rescaling; that is, we instead solve the modified system

$$\begin{pmatrix} A_{11} & \alpha A_{12} \\ \alpha A_{21} & \alpha^2 A_{22} \end{pmatrix} \begin{pmatrix} \mathbf{p} \\ \frac{1}{\alpha}\mathbf{v} \end{pmatrix} = \begin{pmatrix} E\mathbf{u}^* + N^T C Z_1^{-1}\mathbf{w} \\ -\alpha Z_2 Z_1^{-1}\mathbf{w} \end{pmatrix}. \tag{5.36}$$

Theoretically, any non-zero scaling factor α can be plugged into the above system; however, certain values of this parameter may in fact be detrimental. Using the termination condition for iterative solvers as a guide, we propose a simple heuristic for choosing α automatically. In particular, for a linear system $A\mathbf{x} = \mathbf{c}$, an iterative solver refines the current solution until the relative residual, $\|A\mathbf{x} - \mathbf{c}\|/\|\mathbf{c}\|$, falls below some specified threshold. Writing the two parts of (5.25) as $A_1\mathbf{x} = \mathbf{c}_1$ and $A_2\mathbf{x} = \mathbf{c}_2$, we can prevent either $\mathbf{c}_1$ or $\mathbf{c}_2$ from dominating the denominator of residual error by scaling them to be of roughly equal norm; this can be achieved by setting $\alpha = \|\mathbf{c}_1\|/\|\mathbf{c}_2\|$. Choosing α in this manner ensures that both fluid and solid dynamics equations are solved to comparable accuracy by an iterative scheme. Alternatively, it is possible to use a static value of α throughout a simulation. For instance, we found that $\alpha = 0.001$ worked quite well and used it exclusively for all our experiments. However, this setting may need modification if quite different material parameters are used.

While this transformation allows us to acquire an SPD system, we have not yet considered whether doing so might significantly deteriorate sparsity and conditioning as a side-effect. Section 6.2 describes the results of several numerical experiments designed to assess these properties. To summarize our findings, the transformation provides us with an SPD system at a permissibly modest cost to sparsity and conditioning.

5.3 Comparison with Changing to Stress-Like Variables

To acquire an SPD system, Robinson-Mosher et al. [48] instead perform a change of variables, replacing solid velocities with stress-like unknowns. For comparison purposes, we outline this alternative transformation strategy.

Recall the definition of the local stiffness matrix of element t from equation (3.44):

$$K_t = A_D^T A_F^T A_\epsilon^T A_\Psi A_\epsilon A_F A_D, \tag{5.37}$$

The matrix A_Ψ can be written as $L^T L$, where

$$L = \frac{1}{\sqrt{2\mu+\lambda}} \begin{pmatrix} 2\mu+\lambda & \lambda & \lambda & 0 \\ 0 & 2\sqrt{\mu^2+\lambda\mu} & \frac{\lambda\mu}{\sqrt{\mu^2+\lambda\mu}} & 0 \\ 0 & 0 & \mu\sqrt{\frac{4\mu^2+8\lambda\mu+3\lambda^2}{\mu^2+\lambda\mu}} & 0 \\ 0 & 0 & 0 & 2\sqrt{2\mu^2+\lambda\mu}I \end{pmatrix}, \tag{5.38}$$

and I is the 3×3 identity matrix. Letting $R_t = LA_\epsilon A_F A_D$, we can see that $R_t^T R_t = K_t$. It follows that the global stiffness matrix can be factored as $K = R^T R$, where R is formed by stacking R_t:

$$R = \begin{pmatrix} R_1 \\ \vdots \\ R_m \end{pmatrix}. \tag{5.39}$$

Defining $\zeta_M = 1/\Delta t + \tau_M$, $\zeta_K = \Delta t + \tau_K$, we can write (5.17) as

$$\begin{pmatrix} B & N^T C \\ C^T N & -\zeta_M M - \zeta_K R^T R \end{pmatrix} \begin{pmatrix} \mathbf{p} \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} E\mathbf{u}^* \\ \mathbf{w} \end{pmatrix}. \tag{5.40}$$

Isolating $\mathbf{v}$ in the second equation in the same manner as before gives

$$\mathbf{v} = \frac{1}{\zeta_M} M^{-1} \left(C^T N \mathbf{p} - \zeta_K R^T R \mathbf{v} - \mathbf{w} \right). \tag{5.41}$$

We can use this expression to derive

$$E\mathbf{u}^* = B\mathbf{p} + \frac{1}{\zeta_M} N^T C M^{-1} \left(C^T N \mathbf{p} - \zeta_K R^T R \mathbf{v} - \mathbf{w} \right), \tag{5.42}$$

or, equivalently,

$$\left(B + \frac{1}{\zeta_M} N^T C M^{-1} C^T N \right) \mathbf{p} - \frac{\zeta_K}{\zeta_M} N^T C M^{-1} R^T R \mathbf{v} = E\mathbf{u}^* + \frac{1}{\zeta_M} N^T C M^{-1} \mathbf{w}. \tag{5.43}$$

Similarly,

$$\zeta_K R\mathbf{v} = \frac{\zeta_K}{\zeta_M} R M^{-1} \left(C^T N \mathbf{p} - \zeta_K R^T R \mathbf{v} - \mathbf{w} \right), \tag{5.44}$$

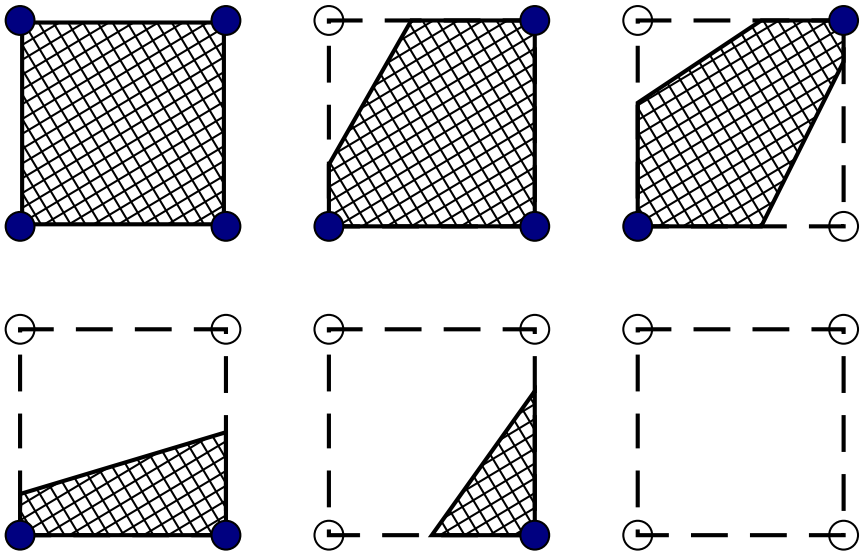

Figure 5.2: The six possible configurations for a face, with the approximate solid region shaded in each.

which can be re-expressed as

$$-\frac{\zeta_K}{\zeta_M} RM^{-1}C^T N\mathbf{p} + \zeta_K \left(I + \frac{\zeta_K}{\zeta_M} RM^{-1}R^T \right) R\mathbf{v} = -\frac{\zeta_K}{\zeta_M} RM^{-1}\mathbf{w}. \tag{5.45}$$

If we define $\mathbf{z} = R\mathbf{v}$, then making this substitution into equations (5.43) and (5.45) yields an SPD system in terms of $\mathbf{p}$ and $\mathbf{z}$. Note that the number of unknowns without the pressure variables has changed from $3n$ to $6m$, where n and m are the number of nodes and elements in the solid mesh, respectively; in most cases this is a significant increase. In addition, our technique is simpler, since it does not call for factorization of the solid stiffness matrix. For a concrete comparison in performance between these two transformations in a simulation scenario, see Section 6.2.

5.4 Algorithm Summary

The augmented pressure projection scheme fits rather conveniently into the previously described fluid simulation loop (Algorithm 3). The main difference within each simulation

Algorithm 4 Coupled Simulation Step

1: update solid positions (using (5.2))
2: advect fluid particles (Section 4.5)
3: advect fluid velocities (Section 4.3)
4: add external accelerations to fluid and solid velocities
5: compute distance fields and cut-cells
6: identify active pressure cells (Section 4.6)
7: perform coupled pressure projection (Chapter 5)
8: extrapolate fluid velocities (Section 4.6)

step (as is evident from Algorithm 4) is computation of cut-cells and solid face areas. The former is done by clipping the solid's surface triangles against the computational grid; for each planar surface region intersecting a cell, we save the area, unit normal oriented into the solid, and barycentric representation of its centroid. For simplicity, solid MAC grid face areas are calculated with the help of the deformable's signed distance field, which is first computed at the nodes of the MAC grid. See Figure 5.2 for the possible configurations and estimated solid region in each case (filled dots indicate that the respective node is inside the solid). Note that a more geometrically faithful approach would compute these face areas based on the exact cut-cell geometry, although we encountered no issues from this mild inconsistency.

Our CPU-based `C++` implementation uses Eigen [29] to handle linear algebra routines and OpenMP to facilitate simple loop parallelization throughout the code. The SPD system is iteratively solved with the conjugate gradient method aided by an incomplete Cholesky preconditioner, with convergence tolerance on the relative residual set to 10^{-10}.

Chapter 6

Results

We conducted several experiments to test and demonstrate our scheme's efficacy. Our findings are presented within the present chapter.

6.1 Animation Examples

For animation examples, all visualizations were obtained by rendering our simulation data in Houdini [55]. Table 6.1 contains simulation resolutions for the described tests. Excluding the stability or conditioning assessments discussed later, the average time per experiment to carry out a single step of simulation ranged between 18 and 59 seconds; of that time, between 3.5 and 39 seconds were spent performing pressure projection. Simulations were performed on a Linux machine (running Ubuntu 16.04) equipped with 16 GB of RAM and AMD FX-8370E clocked at 3.3 GHz. In order to minimize the likelihood of tunneling of liquid particles through solids from overly large timesteps relative to the fluid velocity, we made use of substepping within frames, leading to approximately 4.42 simulation steps per animation frame on average.

6.1.1 Dam Break

Our first scenario consists of an initially vertical column of liquid interacting with a dangling deformable beam pinned at its top. Upon collapsing under gravity, the liquid collides with the beam at which point two-way coupling effects can be witnessed: the fluid's motion is deflected by the solid, and the beam deforms as it is subjected to the accelerating

Table 6.1: Table of simulation resolutions.

Simulation	Grid Size	Solid	
		Nodes	Elements
Dam Break	$100 \times 100 \times 100$	725	2688
Buoyancy	$100 \times 100 \times 100$	481	2000
Compressibility	$100 \times 100 \times 100$	343	1296
Thin Solid 1	$100 \times 100 \times 100$	882	2400
Thin Solid 2	$64 \times 64 \times 64$	2450	6936
Fluid in Sphere	$100 \times 100 \times 100$	992	2948
Light Dino	$100 \times 100 \times 100$	1424	5920

liquid's momentum. After the initial violent interactions, the solid and the liquid settle into a calmer rhythm, gently swaying in tandem. Several frames from this animation are pictured in Figure 6.1.

6.1.2 Buoyancy Example

To demonstrate that our coupling method applies buoyancy forces in a believable manner, we conducted drop experiments with solid balls of different densities. The lightest solid is realistically propelled out of the liquid body, while the heaviest one sinks rapidly to the bottom of the pool. The neutrally buoyant ball (i.e., with rest density equal to the fluid) can be observed to initially maintain its path of motion, then more passively flow with the fluid, and finally sink to the bottom as compression induced by liquid pressure gives it a slightly higher density. Figure 6.3 contains frames from the animation for the lightest solid.

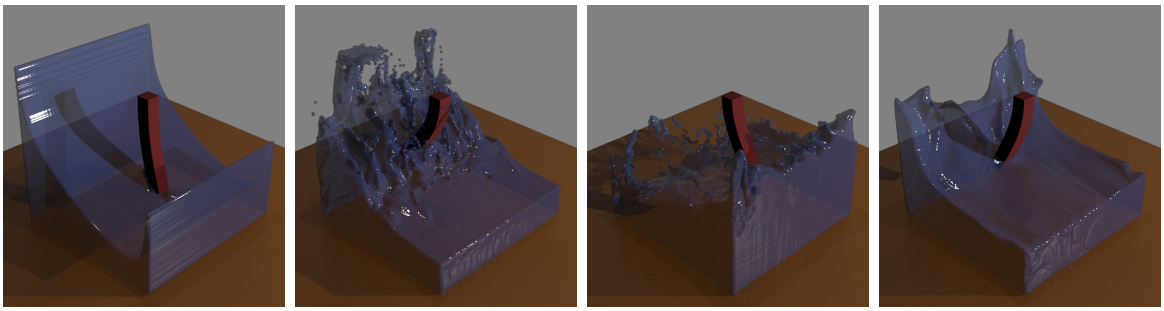

Figure 6.1: A deformable beam pinned at its top acting as an obstacle in a dam break animation.

6.1.3 Solid Compressibility Example

In contrast to the Eulerian coupling scheme of Teng et al. [60], our formulation allows for simulation of compressible solids without special treatment. To showcase this feature, we simulated the expansion of an initially squished cube submerged in a body of water. At the start, this solid is compressed to an eighth of its rest volume; however, elastic forces cause it to rapidly recover its original size. Furthermore, the initially compact solid begins to sink slightly under the influence of gravity, only to be pushed to the surface as expansion lowers its density. Figure 6.2 contains several frames from the described animation.

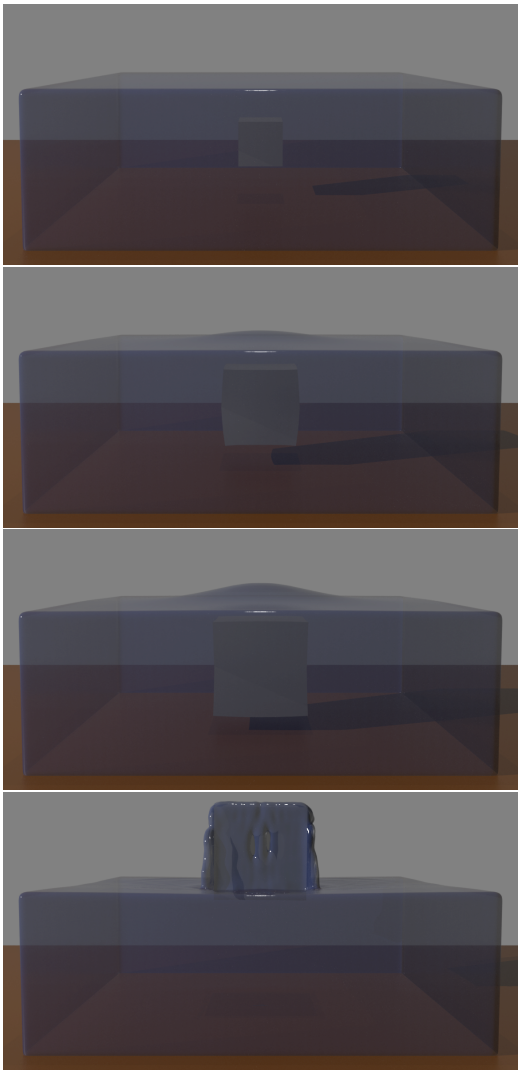

Figure 6.2: A compressed solid is submerged underwater; upon being released, it expands and floats to the surface.

6.1.4 Liquid Supported by a Solid

Two different scenes were set up to illustrate that our method can capture interactions between fluids and deformable solids of modest thickness. For the first scenario, a mass of fluid is dropped on an angled rectangular elastic solid panel that is constrained along two of its sloped sides. Upon contact, the fluid deforms the object as expected. Because the solid is angled, the liquid then starts freely flowing down its slope, showing that the free-slip condition is correctly applied at the solid-fluid interface and no grid artefacts are present. As the fluid flows off the object's surface, the solid elastically rebounds and gradually returns to equilibrium. Refer to Figure 6.4 for a few frames from this scenario.

The second scenario consists of a mass of fluid being dropped on a highly deformable horizontal platform. Rich two-way interactions between the liquid and the solid can be observed: fluid flow stretches and distorts the platform, and, conversely, internal elastic forces of the

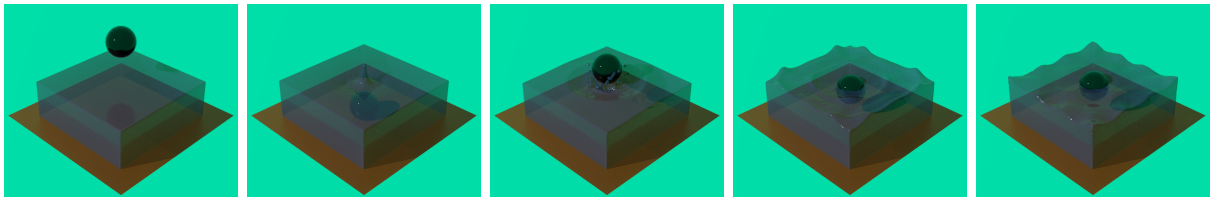

Figure 6.3: A light ball is dropped into a pool of water; as expected, the ball is pushed upwards by buoyancy forces.

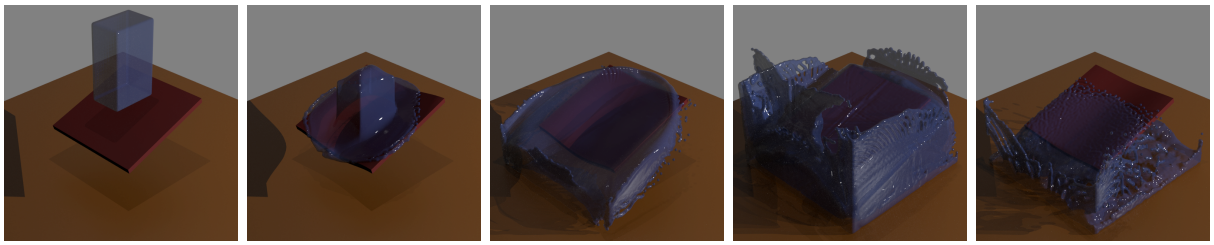

Figure 6.4: Demonstration of free slip: liquid falls atop a sloped deformable platform and flows down.

solid work to undo the induced deformations, further agitating the fluid. As the system settles into equilibrium, the liquid collects into a pool at the center of its deformable enclosure (see Figure 6.9).

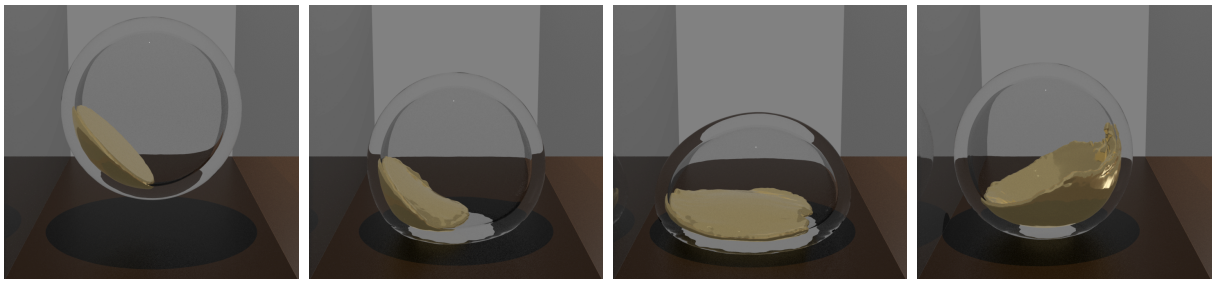

Figure 6.5: Distribution of the fluid within a spherical shell affects the direction of its bounce.

6.1.5 Fluid in a Sphere

In this experiment, a ball of liquid with a high initial velocity is placed inside a hollow deformable sphere. It can be clearly observed that fluid interactions not only locally

distort the solid's shape, but also affect the overall trajectory of the combined system. In particular, the sphere is propelled to one side and bounces off a side wall as a result of the liquid's motion within it (see Figure 6.6). This phenomenon can likewise be seen in Figure 6.5, where an uneven distribution of liquid within the sphere causes it to bounce towards the heavier side.

6.1.6 Light Dino

For our last animation example, we demonstrate a somewhat more complex solid geometry in the form of a light dinosaur-shaped toy (a standard Houdini model) being dropped into a breaking column of water. Two-way interactions can again be observed, with the water being deflected by contact with the solid and the motion of the water distinctly impacting the toy's trajectory. Figure 6.7 contains several frames from this animation.

6.2 SPD System Conditioning

To assess the effect of our SPD transformation technique on conditioning and sparsity of the system, we ran a few instances of our dam break experiment at different grid and mesh resolutions. The simulations were allowed to proceed for 100 frames, and the relevant properties of coefficient matrices were computed and aggregated (condition numbers were estimated using MATLAB's `condest` function [38]). Additionally, time taken to solve the pressure projection system in Eigen was recorded for each formulation. BiCGSTAB in conjunction with an incomplete LU preconditioner [51] was used to solve the indefinite system. We also attempted to solve the indefinite system using MINRES with a block-based preconditioner in the manner suggested by Robinson-Mosher et al. [49], but in our experiments this performed appreciably worse than ILU-preconditioned BiCGSTAB; we

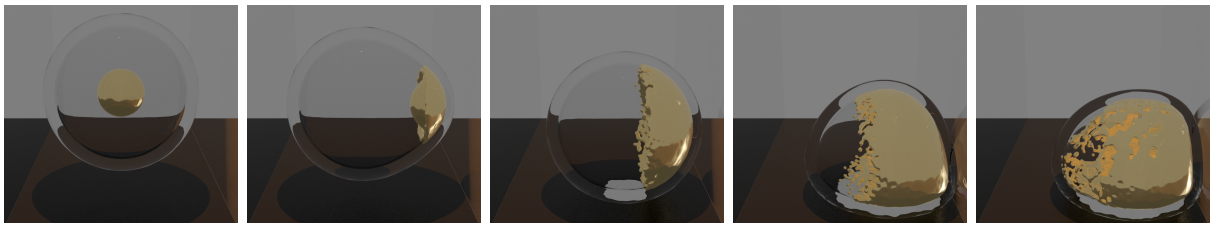

Figure 6.6: A volume of liquid moving within a thick-walled spherical shell affects its overall motion, forcing it to collide with the right wall and bounce.

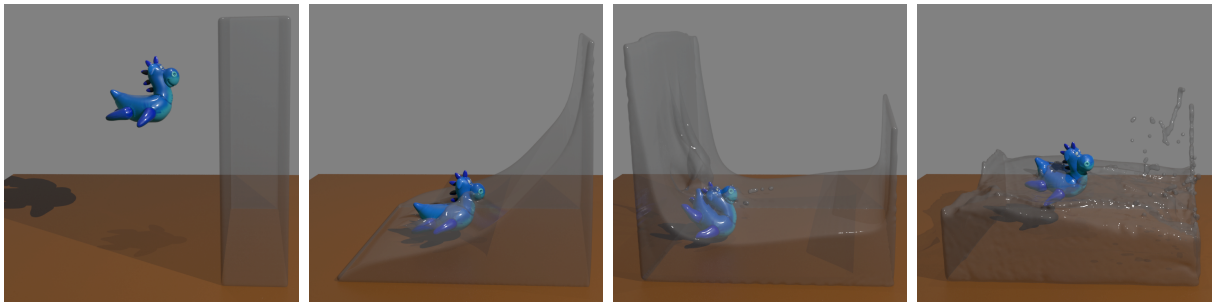

Figure 6.7: A low density elastic dinosaur toy deforming and interacting with liquid under our cut-cell coupling scheme.

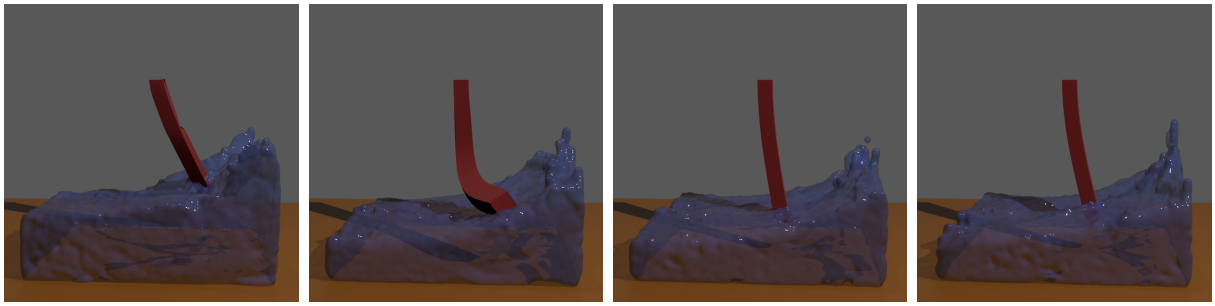

Figure 6.8: Stability assessment; from left to right: decoupled approach, only damping coupled, our method with damping, our method without damping.

therefore present the BiCGSTAB data as our baseline. Our findings are summarized in Table 6.2. The data reveals that the transformation results in a modest rise in matrix density for our test case; in addition, the condition number sees a noticeable increase. (As another point of comparison, a naïve approach to achieve a positive-definite system would be to simply form the normal equations, but this has the significantly worse effect of squaring the condition number.) More importantly, average convergence time for the pressure projection illustrates the advantage of our SPD transformation: the indefinite system generally takes longer to solve. Note also that despite being slightly slower on the coarsest grid, the SPD system can be solved almost twice as fast on the medium grid, and almost four times as fast on the finest grid. That is, the SPD formulation exhibits appreciably better scaling with respect to simulation resolution.

Using the same test setup, we also compared the performance of our SPD transformation technique to the factoring method described in Section 5.3. Our findings are summarized in Tables 6.3 and 6.4; we note that for the coarser solid, the equivalent fluid

grid resolution (with respect to node separation) would be 25^3, and for the finer one, it is 50^3. Hence, our method performs substantially better when the fluid and solid are of comparable resolution. Refinement of the fluid grid for a fixed solid mesh enjoys better scaling under the factoring scheme; conversely, refinement of the solid mesh for a fixed fluid resolution scales better with our proposed transformation. However, thanks to fewer variables in the system, we observe that our method's memory footprint is appreciably smaller, as the number of nonzeros in the pressure projection coefficient matrix is often a small fraction of the factored version.

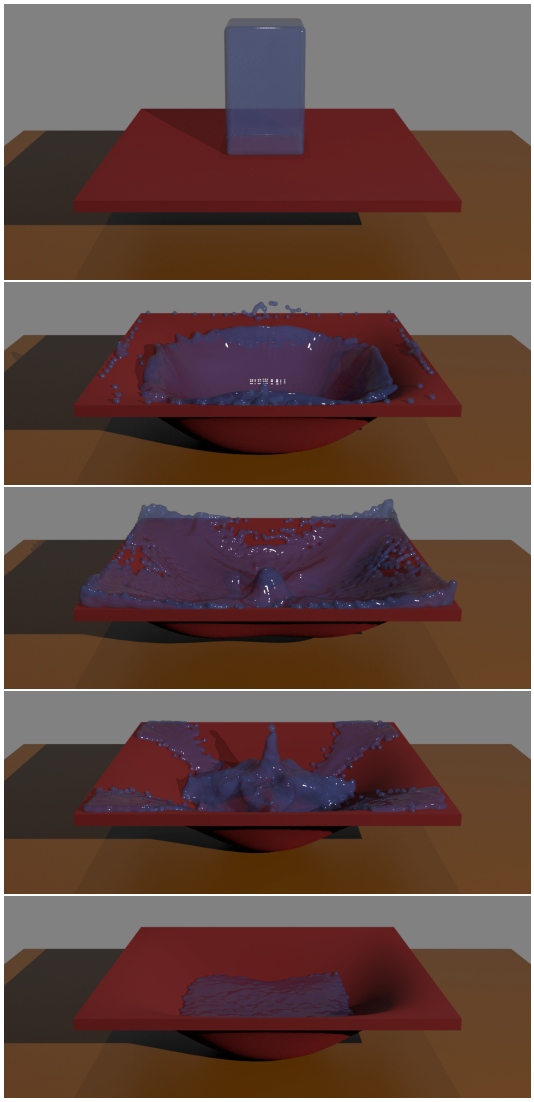

Figure 6.9: A volume of liquid falls onto and comes to rest atop a deformable platform.

6.3 Stability Assessment

The dam break experiment was also used to gauge the effect of strong coupling on stability. To this end, we simulated a low-resolution dam break (35^3 fluid grid, solid with 189 nodes and 480 elements) using several pressure projection schemes: weak (alternating) coupling with the solid, strong coupling with the solid's damping only, and strong coupling of solid damping and elasticity. In order to strain these methods, a stiff but light deformable body was used along with high gravity and relatively large time steps. As expected, the first method was unstable, with the solid undergoing extreme distortion, and the simulation eventually failing. While the simulation was able to complete when only solid damping was coupled with pressure projection, results obtained via this scheme were noticeably unrealistic; specifically, since pressure projection is sequenced after elastic forces, pressure-induced buoyancy inaccurately dominates and incorrectly pushes the stiff solid to the surface. Our proposed fully coupled approach, on the other hand, encountered no dif-

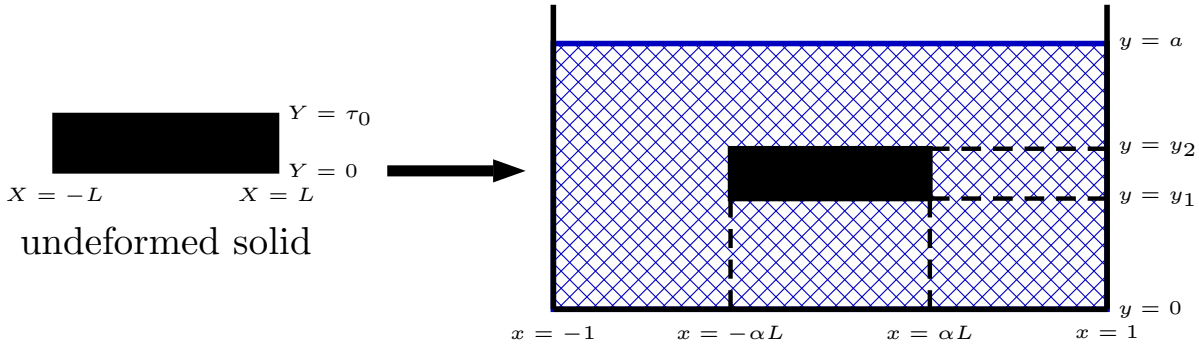

Figure 6.10: Setup for the accuracy assessment.

Table 6.2: Comparison of coefficient matrices for the indefinite and SPD formulations (all entries in this table are averaged over 100 simulation steps); c stands for the condition number.

Fluid Grid Size	Solid Nodes (Elements)	Nonzeros Indefinite	Nonzeros SPD	$\log_{10}$ of c Indefinite	$\log_{10}$ of c SPD	Solve Time (s) Indefinite	Solve Time (s) SPD
40^3	135 (336)	157498	185503	8.347	10.497	1.635	1.813
60^3	352 (1134)	499787	610697	8.579	11.311	16.121	8.292
80^3	725 (2688)	1177419	1439173	8.819	11.497	100.875	28.020

ficulties with this scenario. Additionally, the fully coupled simulation was repeated with zero damping force; once again, our formulation had no problems producing the animation (see Figure 6.8). This result is significant, since the current state-of-the-art method of Robinson-Mosher et al. [47] simultaneously accounts for fluid pressure and solid damping (but not elasticity) and, thus, requires the deformable object to be damped.

6.4 Accuracy Assessment

To numerically assess our method's order of accuracy, we set up a simple scenario and measured how far the velocities deviated from the analytic solution after a single time step. In particular, we analyzed a submerged solid in a pool of liquid, in hydrostatic

Table 6.3: Running time in seconds (total frame time averaged over 100 frames) comparison of our SPD transformation method (data in blue and on the left) to the factored formulation (data in red and on the right).

Fluid Grid Size	Coarse Solid 135 nodes, 336 elements		Fine Solid 725 nodes, 2688 elements	
50^3	5.33	6.69	14.15	27.68
60^3	9.51	9.90	19.76	32.02
70^3	15.75	15.32	29.89	36.98
80^3	23.54	22.21	44.04	47.87
90^3	35.93	31.56	59.09	58.76

Table 6.4: Matrix nonzeros (averaged over 100 frames) comparison of our SPD transformation method (data in blue and on the left) to the factored formulation (data in red and on the right).

Fluid Grid Size	Coarse Solid 135 nodes, 336 elements		Fine Solid 725 nodes, 2688 elements	
50^3	317505	814424	624387	5654643
60^3	501430	998372	807789	5838273
70^3	795668	1218020	1095644	6130135
80^3	1128123	1545752	1425174	6456548
90^3	1565587	1985347	1856102	6890398

equilibrium (refer to Figure 6.10). The deformation map for this solid is chosen to be

$$\phi(X,Y) = \begin{pmatrix} \alpha X \\ \beta Y^2 + \gamma Y + y_1 \end{pmatrix} \tag{6.1}$$

and its first Piola-Kirchoff stress tensor can be derived from (3.3) and (3.8):

$$P(X,Y) = \begin{pmatrix} (2\mu+\lambda)(\alpha-1) + \lambda(2\beta Y + \gamma - 1) & 0 \\ 0 & (2\mu+\lambda)(2\beta Y+\gamma-1) + \lambda(\alpha-1) \end{pmatrix}, \tag{6.2}$$

where

$$\beta = -\frac{\alpha(y_2-y_1)\rho_f g}{2\tau_0(2\mu+\lambda)}, \tag{6.3}$$

$$\gamma = \frac{y_2-y_1}{\tau_0} - \beta\tau_0. \tag{6.4}$$

Table 6.5: First accuracy assessment ($\mu = 20, \lambda = 10, L = \tau_0 = 0.4, \alpha = 0.9, y_1 = 0.205, y_2 = 0.595$).

Δx	Pressures		Solid Velocities		Liquid Velocities	
	Error	Order	Error	Order	Error	Order
0.2000	4.19×10^{-2}		5.68×10^{-3}		3.75×10^{-3}	
0.1000	4.22×10^{-2}	-0.01	4.20×10^{-3}	0.44	2.66×10^{-3}	0.49
0.0500	1.84×10^{-2}	1.20	1.93×10^{-3}	1.12	2.31×10^{-3}	0.20
0.0250	6.95×10^{-3}	1.41	9.75×10^{-4}	0.98	1.14×10^{-3}	1.01
0.0125	1.15×10^{-3}	2.59	2.72×10^{-4}	1.84	3.01×10^{-4}	1.93

The above formula for β ensures that the solid's internal forces ($\nabla \cdot P$) cancel out the effect of gravity, and γ is set to satisfy the boundary condition ($Y = \tau_0$ maps to $y = y_2$). In these expressions, ρ_f denotes the fluid density, and g is acceleration due to gravity (acting along the vertical axis). For the liquid, the pressure field is simply

$$p(x, y) = \rho_f g(y - a). \tag{6.5}$$

It is not difficult to check that the solid and fluid are indeed in hydrostatic equilibrium, assuming the following tractions are applied to the solid on the left, right, top, and bottom, respectively:

$$\begin{aligned} T_L(Y) = &-(2\mu + \lambda)(\alpha - 1) - \lambda(2\beta Y + \gamma - 1) \\ &- \rho_f g(\beta Y^2 + \gamma Y + y_1 - a)(2\beta Y + \gamma), \end{aligned} \tag{6.6}$$

$$\begin{aligned} T_R(Y) = &(2\mu + \lambda)(\alpha - 1) + \lambda(2\beta Y + \gamma - 1) \\ &+ \rho_f g(\beta Y^2 + \gamma Y + y_1 - a)(2\beta Y + \gamma), \end{aligned} \tag{6.7}$$

$$T_T(X) = (2\mu + \lambda)(2\beta\tau_0 + \gamma - 1) + \lambda(\alpha - 1) + \alpha\rho_f g(y_2 - a), \tag{6.8}$$

$$T_B(X) = -(2\mu + \lambda)(\gamma - 1) - \lambda(\alpha - 1) - \alpha\rho_f g(y_1 - a). \tag{6.9}$$

Since these expressions are arrived at from the first Piola-Kirchoff stress, we highlight that these tractions must be integrated with respect to the reference configuration. For completeness, we state that in order for the densities to match, the solid's initial density field is constant and given by

$$\rho_s = \frac{\alpha(y_2 - y_1)\rho_f}{\tau_0}. \tag{6.10}$$

Our accuracy experiments yielded inconsistent results. Refer to Tables 6.5 and 6.6 for some of our data (for all of our trials, we used $\rho_f = 1$, $g = -1$, $a = 0.8$, $\Delta t = 0.01$ and

Table 6.6: Second accuracy assessment ($\mu = 100, \lambda = 100, L = \tau_0 = 0.4, \alpha = 1.1, y_1 = 0.18, y_2 = 0.62$).

Δx	Pressures		Solid Velocities		Liquid Velocities	
	Error	Order	Error	Order	Error	Order
0.2000	9.19×10^{-2}		3.92×10^{-3}		3.02×10^{-3}	
0.1000	3.35×10^{-2}	1.46	1.37×10^{-3}	1.51	1.06×10^{-3}	1.51
0.0500	6.01×10^{-3}	2.48	3.25×10^{-4}	2.07	2.08×10^{-4}	2.35
0.0250	7.30×10^{-3}	-0.28	4.74×10^{-4}	-0.55	7.57×10^{-4}	-1.86
0.0125	1.21×10^{-3}	2.59	1.07×10^{-4}	2.15	1.92×10^{-4}	1.98

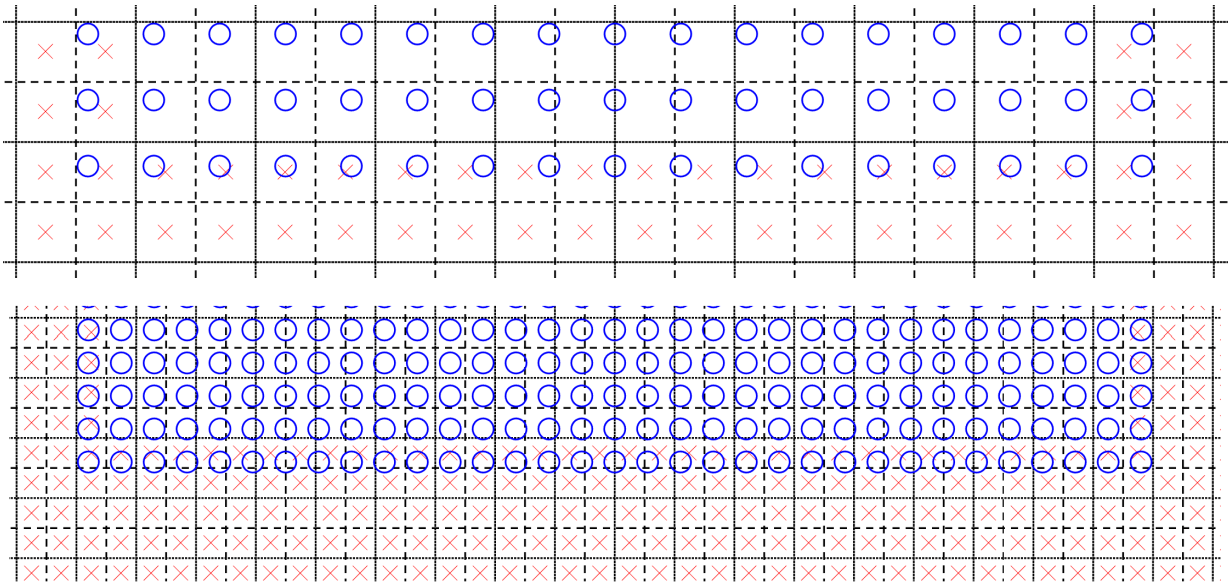

Figure 6.11: Refinement of the computational grid increases the distance between pressure samples (given by $\times$) and solid boundary (nodes given by $\circ$).

the ∞-norm was utilized for error computations). In certain cases, the coupling schemes exhibits first order accuracy, but sometimes the error scales better than linearly with respect to spatial discretization. In yet other scenarios, refinement of the computational grids does not reduce the errors at all; for these cases, we noticed that the distance between solid nodes and pressure samples actually increased in the higher-resolution discretization (see Figure 6.11). Thus, we suspect that this behaviour may be due to our assumption that pressure is constant within each fluid cell.

Chapter 7

Conclusions

We have presented a novel approach to simulating two-way coupling between fluids and deformable bodies. Compared to previous solutions, our cut-cell method facilitates accurate free-slip boundary condition enforcement at the true solid-fluid interface without adding new degrees of freedom. Due to a fully implicit treatment of solid dynamics as part of the pressure projection, our technique also enjoys better stability properties. Additionally, an efficient symmetric positive-definite transformation scheme was introduced that allows us to convert the symmetric indefinite projection system into a form that is appreciably faster to solve numerically.

7.1 Future Work

We conclude by listing possible avenues for future research. With respect to simulation of thin solids, it was established that our method is capable of producing animations if the object's thickness is greater than a grid cell width or two. However, prevention of fluid flow through the solid is much more difficult when it has a very small or zero thickness (e.g., cloth); furthermore, differences in pressure and velocity on opposite sides of the object must be taken into account. We expect that our method could be combined with the techniques proposed by Azevedo et al. [2] to allow for efficient simulation of scenarios involving such thin solids or shells. We assumed a simple isotropic linear elastic corotational model that yields realistic results for many deformable objects, but the dynamics of more complex materials cannot be adequately captured with this approach. To properly treat nonlinear constitutive models within our framework, a linearization of the solid forces or a full nonlinear solve would likely be required. Improved numerical linear algebra solvers

tailored to our SPD system, such as multigrid schemes, could benefit our timing results. To allow the liquid to freely separate from solids, as originally proposed by Batty et al. [5], one could explore extending our method with inequality boundary conditions leading to an LCP system. Finally, we treated contact between the deformable solid and the domain boundaries with a simple explicit correction; it could also be valuable to incorporate implicit contact response into our system. For configurations involving multiple deformable objects and complex contact, we expect that this extension would further improve stability and accuracy.

References

[1] Nadir Akinci, Jens Cornelis, Gizem Akinci, and Matthias Teschner. Coupling elastic solids with smoothed particle hydrodynamics fluids. *Computer Animation and Virtual Worlds*, 24(3-4):195–203, 2013.

[2] Vinicius C Azevedo, Christopher Batty, and Manuel M Oliveira. Preserving geometry and topology for fluid flows with thin obstacles and narrow gaps. *ACM Transactions on Graphics (TOG)*, 35(4):97, 2016.

[3] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 43–54. ACM, 1998.

[4] George Keith Batchelor. *An Introduction to Fluid Dynamics*. Cambridge university press, 2000.

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

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

[7] Markus Becker and Matthias Teschner. Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 209–217. Eurographics Association, 2007.

[8] David E Breen, Donald H House, and Michael J Wozny. Predicting the drape of woven cloth using interacting particles. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pages 365–372. ACM, 1994.

[9] Robert Bridson. *Fluid Simulation for Computer Graphics*. CRC Press, 2015.

[10] Tyson Brochu, Christopher Batty, and Robert Bridson. Matching fluid simulation elements to surface geometry and topology. In *ACM SIGGRAPH 2010 Papers*, pages 47:1–47:9, 2010.

[11] Mark Carlson, Peter J Mucha, and Greg Turk. Rigid fluid: animating the interplay between rigid bodies and fluid. *ACM Transactions on Graphics (TOG)*, 23(3):377–384, 2004.

[12] Jim X Chen and Niels da Vitoria Lobo. Toward interactive-rate simulation of fluids with moving obstacles using navier-stokes equations. *Graphical Models and Image Processing*, 57(2):107–116, 1995.

[13] Nuttapong Chentanez, Tolga G Goktekin, Bryan E Feldman, and James F O'Brien. Simultaneous coupling of fluids and deformable bodies. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 83–89. Eurographics Association, 2006.

[14] Pascal Clausen, Martin Wicke, Jonathan R Shewchuk, and James F O'brien. Simulating liquids and solid-liquid interactions with lagrangian meshes. *ACM Transactions on Graphics (TOG)*, 32(2):17, 2013.

[15] Mathieu Desbrun and Marie-Paule Gascuel. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Computer Animation and Simulation*, pages 61–76. Springer, 1996.

[16] Essex Edwards and Robert Bridson. Detailed water with coarse grids: combining surface meshes and adaptive discontinuous galerkin. *ACM Transactions on Graphics (TOG)*, 33(4):136, 2014.

[17] 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. In *ASME/JSME 2003 4th Joint Fluids Summer Engineering Conference*, pages 337–342. American Society of Mechanical Engineers, 2003.

[18] Douglas Enright, Frank Losasso, and Ronald Fedkiw. A fast and accurate semi-lagrangian particle level set method. *Computers & Structures*, 83(6):479–490, 2005.

[19] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 736–744. ACM, 2002.

[20] Olaf Etzmuß, Michael Keckeisen, and Wolfgang Straßer. A fast finite element solution for cloth modelling. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, pages 244–251. IEEE, 2003.

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

[22] Nick Foster and Ronald Fedkiw. Practical animation of liquids. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 23–30. ACM, 2001.

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

[24] Alain Fournier and William T Reeves. A simple model of ocean waves. *ACM SIGGRAPH Computer Graphics*, 20(4):75–84, 1986.

[25] TF Gast and C Schroeder. Optimization integrator for large time steps. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 31–40. Eurographics Association, 2014.

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

[27] Frédéric Gibou and Chohong Min. Efficient symmetric positive definite second-order accurate monolithic solver for fluid/solid interactions. *Journal of Computational Physics*, 231(8):3246–3263, 2012.

[28] Eran Guendelman, Andrew Selle, Frank Losasso, and Ronald Fedkiw. Coupling water and smoke to thin deformable and rigid shells. *ACM Transactions on Graphics (TOG)*, 24(3):973–981, 2005.

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

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

[31] Florian Hecht, Yeon Jin Lee, Jonathan R Shewchuk, and James F O'Brien. Updated sparse cholesky factors for corotational elastodynamics. *ACM Transactions on Graphics (TOG)*, 31(5):123, 2012.

[32] Ben Houston, Chris Bond, and Mark Wiebe. A unified approach for modeling complex occlusions in fluid simulations. In *ACM SIGGRAPH 2003 Sketches & Applications*, pages 1–1. ACM, 2003.

[33] Michael Kass and Gavin Miller. Rapid, stable fluid dynamics for computer graphics. In *ACM SIGGRAPH Computer Graphics*, volume 24, pages 49–57. ACM, 1990.

[34] Bryan M Klingner, Bryan E Feldman, Nuttapong Chentanez, and James F O'brien. Fluid animation with dynamic meshes. *ACM Transactions on Graphics (TOG)*, 25(3):820–825, 2006.

[35] David IW Levin, Joshua Litven, Garrett L Jones, Shinjiro Sueda, and Dinesh K Pai. Eulerian solid simulation with contact. *ACM Transactions on Graphics (TOG)*, 30(4):36, 2011.

[36] Chia-Chiao Lin and Lee A Segel. *Mathematics Applied to Deterministic Problems in the Natural Sciences*. SIAM, 1988.

[37] Wenlong Lu, Ning Jin, and Ronald Fedkiw. Two-way coupling of fluids to reduced deformable bodies. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 67–76. Eurographics Association, 2016.

[38] Mathworks. Matlab. https://www.mathworks.com/, 2017.

[39] Nelson L Max. Vectorized procedural models for natural terrain: Waves and islands in the sunset. *ACM SIGGRAPH Computer Graphics*, 15(3):317–324, 1981.

[40] Marek Krzysztof Misztal, Kenny Erleben, Adam Bargteil, Jens Fursund, Brian Bunch Christensen, Jakob Andreas Bærentzen, and Robert Bridson. Multiphase flow of immiscible fluids on unstructured moving meshes. *IEEE Transactions on Visualization and Computer Graphics*, 20(1):4–16, 2014.

[41] 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*, pages 154–159. Eurographics Association, 2003.

[42] Matthias Müller and Markus Gross. Interactive virtual materials. In *Proceedings of Graphics Interface 2004*, pages 239–246. Canadian Human-Computer Communications Society, 2004.

[43] Yen Ting Ng, Chohong Min, and Frédéric Gibou. An efficient fluid–solid coupling algorithm for single-phase flows. *Journal of Computational Physics*, 228(23):8807–8829, 2009.

[44] Hidehiko Okabe, Haruki Imaoka, Takako Tomiha, and Haruo Niwaya. Three dimensional apparel cad system. In *ACM SIGGRAPH Computer Graphics*, volume 26, pages 105–110. ACM, 1992.

[45] Vito Pasquariello, Georg Hammerl, Felix Örley, Stefan Hickel, Caroline Danowski, Alexander Popp, Wolfgang A Wall, and Nikolaus A Adams. A cut-cell finite volume–finite element coupling approach for fluid–structure interaction in compressible flow. *Journal of Computational Physics*, 307:670–695, 2016.

[46] Nick Rasmussen, Douglas Enright, Duc Nguyen, Sebastian Marino, Nigel Sumner, Willi Geiger, Samir Hoon, and Ronald Fedkiw. Directable photorealistic liquids. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 193–202. Eurographics Association, 2004.

[47] Avi Robinson-Mosher, R Elliot English, and Ronald Fedkiw. Accurate tangential velocities for solid fluid coupling. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 227–236. ACM, 2009.

[48] Avi Robinson-Mosher, Craig Schroeder, and Ronald Fedkiw. A symmetric positive definite formulation for monolithic fluid structure interaction. *Journal of Computational Physics*, 230(4):1547–1566, 2011.

[49] 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 Transactions on Graphics (TOG)*, 27(3):46, 2008.

[50] Doug Roble, Nafees bin Zafar, and Henrik Falt. Cartesian grid fluid simulation with irregular boundary voxels. In *ACM SIGGRAPH 2005 Sketches*, page 138. ACM, 2005.

[51] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2003.

[52] Lee A Segel and GH Handelman. *Mathematics Applied to Continuum Mechanics*. SIAM, 2007.

[53] Rahul Sheth, Wenlong Lu, Yue Yu, and Ronald Fedkiw. Fully momentum-conserving reduced deformable bodies with collision, contact, articulation, and skinning. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 45–54. ACM, 2015.

[54] Eftychios Sifakis and Jernej Barbic. Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH 2012 Courses*, page 20. ACM, 2012.

[55] Side Effects Software. Houdini. https://www.sidefx.com/, 2017.

[56] Barbara Solenthaler, Jürg Schläfli, and Renato Pajarola. A unified particle model for fluid–solid interactions. *Computer Animation and Virtual Worlds*, 18(1):69–82, 2007.

[57] Fotis Sotiropoulos and Xiaolei Yang. Immersed boundary methods for simulating fluid–structure interaction. *Progress in Aerospace Sciences*, 65:1–21, 2014.

[58] Jos Stam. Stable fluids. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 121–128. ACM Press/Addison-Wesley Publishing Co., 1999.

[59] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)*, 32(4):102, 2013.

[60] Yun Teng, David IW Levin, and Theodore Kim. Eulerian solid-fluid coupling. *ACM Transactions on Graphics (TOG)*, 35(6):200, 2016.

[61] Joseph Teran, Sylvia Blemker, V Hing, and Ronald Fedkiw. Finite volume methods for the simulation of skeletal muscle. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 68–74. Eurographics Association, 2003.

[62] Demetri Terzopoulos and Kurt Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, 1988.

[63] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. *ACM SIGGRAPH Computer Graphics*, 21(4):205–214, 1987.

[64] Daniel Weber, Johannes Mueller-Roemer, André Stork, and Dieter Fellner. A cut-cell geometric multigrid poisson solver for fluid simulation. *Computer Graphics Forum*, 34(2):481–491, 2015.

[65] Gary D Yngve, James F O'Brien, and Jessica K Hodgins. Animating explosions. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, pages 29–36. ACM Press/Addison-Wesley Publishing Co., 2000.

[66] Omar Zarifi and Christopher Batty. A positive-definite cut-cell method for strong two-way coupling between fluids and deformable bodies. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, page 7. ACM, 2017.