COUPLED SIMULATION OF DEFORMABLE SOLIDS, RIGID BODIES, AND FLUIDS WITH SURFACE TENSION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Craig Schroeder
May 2011

Abstract

This thesis considers the numerical simulation of a variety of phenomena, particularly rigid bodies, deformable bodies, and incompressible fluids. We consider each of these simulations types in isolation, addressing challenges specific to each. We also address the problem of monolithic two-way coupling of each of these phenomena.

First we address the stability of rigid body simulation with large time steps. We develop an energy correction for orientation evolution and another correction for collisions. In practice, we have found these two corrections to be sufficient to produce stable simulations. We also explore a simple scheme for rigid body fracture that is as inexpensive as prescoring rigid bodies but more flexible.

Next we develop a method for simulating deformable but incompressible solids. Many constitutive models for deforming solids, such as the neo-Hookean model, break down in the incompressible limit. Simply enforcing incompressibility per tetrahedron leads to locking, where the mesh non-physically resists deformation. We present a method that uses a pressure projection similar to what is commonly used to simulate incompressible solids and apply it to deforming solids. We also address the complications that result from the interaction of this new force with contacts and collisions.

Then, we turn to two coupling problems. The first problem is to couple deformable bodies to rigid bodies. We develop a fully-unified time integration scheme, where individual steps like collisions and contact are each fully two-way coupled. The resulting coupling scheme is monolithic with fully coupled linear systems. This leads to

a robust and strongly coupled simulation framework. We use state-of-the-art integrators for rigid bodies and deformable bodies as the basis for the coupling scheme and maintain the ability to handle other phenomena, such as articulation and controllers on the rigid bodies and incompressibility on the deformable bodies.

We follow this up by developing a scheme for coupling solids to incompressible fluids. The method handles both deformable bodies and rigid bodies. Unlike many existing methods for fluid structure interaction, which often typically lead to indefinite linear systems, the developed scheme results in a symmetric and positive definite (SPD) linear system. In addition to strongly coupling solids and fluids, the method also strongly couples viscosity with fluid pressure. This allows it to accurately treat simulations with high viscosity or where the primary coupling between solid and fluid is through fluid viscosity rather than fluid pressure. The method can be interpreted as a means of converting symmetric indefinite KKT systems with a particular form into SPD systems.

Finally, we propose a method for applying implicit Lagrangian forces to an Eulerian Navier-Stokes simulation. We utilize the SPD framework to produce an SPD system with these implicit forces. We use this method to apply implicit surface tension forces. This implicit surface tension treatment reduces the tight time step restriction that normally accompanies explicit treatments of surface tension.

Acknowledgments

I would like to thank my mother, Irene, for being encouraging and supportive of my education, interests, and ambitions. I would like to thank my sister Diane for motivating me to learn more, lending a hand whenever I needed help, and making Stanford feel a little closer to home. I would also like to thank my sister Linda and my brother Brian.

I would like to thank my advisor Ronald Fedkiw for his encouragement, advice, and sense of direction. He has taught me how to do good research and work hard to achieve my goals. He has always been there with perspective, insight, and ideas whenever I have found myself lost without any clear direction. I would also like to thank Ron for the many opportunities he has provided for me.

Ron's research lab has been an exciting and interesting place to work. I would like to thank Geoffrey Irving for sharing his perspective on what a transpose is. I would like to thank Tamar Shinar for the things that she has taught me and for the advice that she has given me. I would like to thank Jonathan Su for keeping the lab fun. I would like to acknowledge my collaborators Geoffrey Irving, Tamar Shinar, Jonathan Su, Wen Zheng, Avi Robinson-Mosher, Michael Lentine, Jón Grétarsson, and Nipun Kwatra. There have been many others in Ron's lab whom I have had the opportunity to interact with whom I would also like to acknowledge: Robert Strzodka, Jeong-Mo Hong, Rachel Weinstein, Frank Losasso Petterson, Eftychis Sifakis, Andrew Selle, Jerry Talton, Kevin Der, Elliot English, and Mridul Aanjaneya. I would also like to thank Joseph Teran for helping my out with my plans for after graduation.

I would also like to thank Stanford University and the computer science department for fostering an excellent learning environment filled with intelligent, motivated, and interesting people. I would also like to thank Sequoia Capital for the Stanford Graduate Fellowship that has funded me for much of my graduate career.

I have worked at Pixar for most of the time I have been at Stanford. Pixar is a great company to work for, and I have met many wonderful people there. I would like in particular to thank John Anderson for all that he has taught me and for his unique perspective on things. I would also like to thank Mark Meyer, Tony Derose, Kinrick Kin, and everyone else in Pixar Research for making my time at Pixar fun.

I would like to thank my reading committee, Ron, Oussama Khatib, and John Anderson, and the rest of my thesis committee Adrian Lew and Vladlen Koltun. Their feedback and suggestions have been helpful in improving this thesis.

Finally I would like to thank Quinn Finkel for being such a good friend. I would also like to thank Hal Finkel and Pinkesh Patel for the many interesting conversations we have had and all the things we have done together.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Numerical simulation of the laws of physics has important applications in many fields. These applications heavily dictate the type of phenomena simulated and the relative importance of accuracy and stability. On one side of the spectrum is the entertainment industry, where the primary concern is results that are visually plausible. The examples being simulated are often complex and involve diverse phenomena like rigid bodies, deformable solids, and fluids. Since accuracy is of only limited importance, the simulation time step size is often limited only by the stability and robustness of the underlying methods. The requirement of accuracy is also reasonably relaxed in cases like collisions where chaos and inaccurate models limit the meaningfulness of simulation accuracy. At the other end of the spectrum are engineering applications where accurate predictions are required. Such simulations are useful to the extent that they accurately predict what will happen in the real world. In these cases, the time step size is often limited by accuracy requirements. While stability is still important, more emphasis is placed on accuracy and convergence.

Applications for full coupling between different phenomena are numerous and of great interest. Ideally, numerical methods would be developed that perform well when applied to a wide range of phenomena. The reality of the situation, however, is that methods are designed specifically to simulate only a few phenomena. One may

approach coupling by treating different phenomena using the methods developed for one of the individual types. Thus one might couple deformable solids to fluids by treating a deforming body as a viscoelastic fluid or couple rigid bodies to deformable bodies by treating rigid bodies as extremely stiff deforming bodies. While this strategy makes coupling straightforward, it does not permit the use of state-of-the-art methods for each of the individual simulation types.

A simple approach to coupling is to use a partitioned method. Partitioned methods work by building subsystems to simulate each individual type of phenomenon. Then, each subsystem is evolved one time step using the other systems as boundary conditions. This approach is very convenient, since the individual subsystems may be developed independently and may even be black box solvers. This only produces a weak form of coupling with limited information transfer between the different subsystems in a single time step. This problem may be alleviated by iterating the system, but this can be prohibitively expensive.

An alternative approach is monolithic coupling, where each of the individual subsystems are unified into a single method with a coupled time integration scheme and coupled implicit systems to solve. The coupled implicit system produces very strong coupling. While these methods are significantly more difficult to develop, they often have more favorable stability characteristics.

This thesis considers the simulation of a diverse range of phenomena with special emphasis on stability and monolithic coupling of different phenomena. Chapter 2 (based on [144]) makes a number of improvements to the stability of rigid body simulation with large time step sizes. Chapter 3 (based on [80]) develops a method for robustly and stably simulating incompressible deformable bodies in the presence of collisions with other objects. Chapter 4 (based on [134]) presents a method for fully two-way monolithic coupling of rigid bodies and deformable bodies. The method supports collisions, contacts, articulation, controllers, incompressibility, and bindings. Chapter 5 (based on [124]) presents a novel method for monolithic fluid structure interaction. The method produces a symmetric positive definite system which is sparse to apply

and supports both deformable bodies and rigid bodies. Finally, Chapter 6 (based on [128]) develops a method for applying Lagrangian forces to an Eulerian simulation and uses it to ease the tight surface tension time step restriction.

Chapter 2

Frame Rate Rigid Bodies

Rigid bodies under the influence of gravity and collisions do not have a stability time step restriction per se, even when evolved using explicit methods. They do, however, have a time step restriction based on accuracy. In practice these accuracy problems can cause significant increases in energy and highly nonphysical behavior, and we regard such simulations as unstable even though the extent of the problem typically remains bounded. This chapter considers two mechanisms by which a significant nonphysical energy increase is observed to occur in practice. In each case, the source of the error is localized to a single step of the algorithm, and the errors are detected and corrected by carefully considering changes in kinetic energy. We also consider a new approach to fracture that is similar to but more flexible than prescoring. Fracture tends to create sliver bodies, which are particularly difficult to simulate accurately. This chapter is based on the work published in [144].

2.1 Introduction

Rigid body dynamics have been of great interest to the graphics community beginning with the early work of [66, 103, 6, 7, 8], and optimizing these algorithms to be more

efficient was also of early interest, see e.g. [9]. More recently, various authors have revisited rigid body simulation, focusing on a variety of phenomena [64, 87, 88]. There has also been interest in fracture [5] and magnetism [153]. Though all these rigid body techniques have been highly successful in the visual effects industry, the amount of computational power they require has prevented their use in real-time applications. However, when one examines today's real-time rendering systems it is clear that modern and next-generation hardware provides for the rendering of incredibly complex and detailed scenes in real-time, implying that significant computational resources are available. While real-time rendering systems must only compute once per frame, making all resources available for handling higher scene complexity, physics engines typically perform many small time steps per frame, trading spatial complexity for temporal complexity. Our stated goal is to simulate rigid bodies at the frame rate, taking one time step per frame so that we can simulate as detailed a scene as possible. While any basic simulation engine could be used as a starting point for this, we chose [64], emphasizing that many of our ideas for robustly handling contacts, collisions, rigid body evolution, etc. could be adapted to other algorithmic frameworks with relatively little effort.

Numerical simulations are typically designed to converge to the correct solution as the time step goes to zero. Moreover, shrinking the time step improves stability, robustness, and accuracy. Therefore, constraining our simulations to go at the frame rate will pose problems for almost every part of the algorithm. [32] discusses some known flaws in state-of-the-art collision models including common misconceptions, mentioning for example that the coefficient of restitution can be greater than one in frictional collisions. [143] discusses non-unique solutions and the impossibility of predicting which solution occurs in practice. We further illustrate that the standard evolution equations can lead to overall energy gain. Obviously, energy should be conserved if not dissipated due to damage and heat, and since this effect is highly exacerbated using large time steps (i.e. the 30 Hz frame rate), we propose a novel method for clamping collisions in order to guarantee that energy does not increase. We note there has been work on energy conservation with respect to collisions and

contact in *deforming* bodies, as in [3]. Whereas one might debate the correct behavior, it is pretty clear from the aforementioned references that this is not yet understood, and clamping the energy at least leads towards better plausibility in the sense of [13]. Similar ideas are needed for processing contact.

Besides the known problems with contact and collision, rigid body simulations are prone to accuracy errors during evolution such as energy increase. These errors can be quite unacceptable for bodies that are rotating quickly or have ill-conditioned inertia tensors, such as those often introduced during simulations of fracturing bodies. These problems are well known and have been addressed by several authors, including [161], who propose a rather complicated analytic solution to the problem. [162] instead focused on the conservation of the first integrals of the system. In the mechanics literature, there has been work on energy and momentum conserving rigid body dynamics, such as one of the algorithms in [140], but this algorithm requires the iterative solution of a fully nonlinear equation and therefore does not fit well with our goal of efficient simulation. In the same vein we propose a novel algorithm for conserving kinetic energy as rigid bodies rotate. In some sense this focus on energy conservation to provide better stability is analogous to variational integration, see e.g. [89, 24, 91].

Both virtual environment designers and video game developers have found that their users desire dynamic environments, and one of the simplest ways to provide is through rigid body simulation. Whereas a number of systems provide basic rigid simulation in real time, much of the recent interest is focused on destruction scenarios. In fact, fracture has been of great interest to the graphics community for some time [111] and has regained recent popularity via [114, 173, 113]. Other notable works include the real time work of [104], the virtual node algorithm for decomposing meshes [101], and the fracturing of rigid bodies [5]. One can envision fracturing an object in two specific ways. The object can be prescored and subsequently broken apart based on a variety of rules, or one can compute the fracture dynamically using finite element analysis to determine internal stress. Prescoring the material is faster and thus desirable for real-time simulations; however, a great deal of realism is lost because the object does

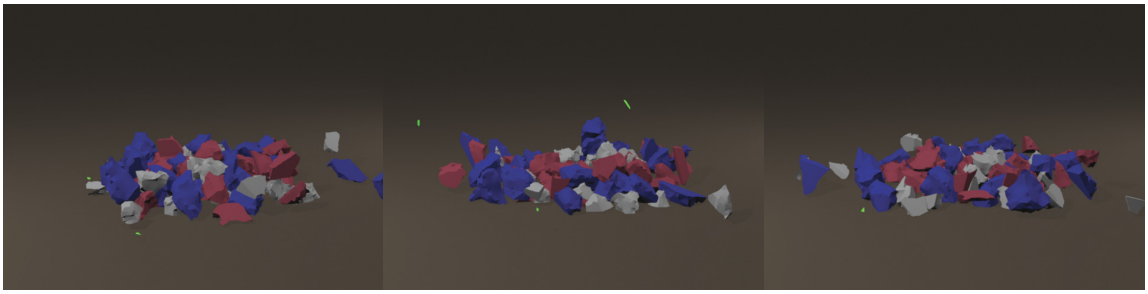


Figure 2.1: The images above show a prescored pillar being dropped onto the ground with (left) a small time step, (center) a frame-rate time step with no energy conservation, (right) a frame rate time step with energy conservation. During both the small time step simulation and the frame-rate simulation with energy conservation, the small shards highlighted in green behave as expected. However, in the frame-rate simulation with no energy conservation, the green shards bounce non-physically high into the air.

not fracture based on the point of impact. We propose a novel method to *prescore all of space* and then subsequently apply this prescoring in the specific location of impact to generate the fracture. This avoids the need for expensive algorithms that compute the stress and fracture dynamically while still breaking the objects based on the point of impact with the efficiency of prescoring. A further benefit of our approach is that it gives the artist more control over the fracture patterns without constraining where the fracture is initiated, i.e. where the collision takes place. (We refer the reader to a rather interesting paper on generating fracture patterns [78].)

2.2 Rigid Body Evolution

Simple translation through space is trivially integrated accurately at any time step, whereas orientation is non-linear and more complex. Typically one solves an ordinary differential equation using a scheme written to conserve angular momentum exactly, but kinetic energy is not necessarily conserved. In certain scenarios, a small time step is needed to guarantee accuracy, and a time step equal to the frame rate can cause an unacceptable gain in energy. For well-conditioned bodies with well-behaved

inertia tensors and relatively slow rotations, the second order evolution suggested in [134] and described in more detail in Section 4.4.1 adequately prevents energy growth. However, for poorly-conditioned and rapidly rotating bodies evolved at the frame rate, we have observed energy increases of over an order of magnitude in a single time step even with higher order evolution. This can lead to significant and noticeable artifacts, such as those seen in Figure 2.1.

In order to properly conserve kinetic energy during evolution one must adjust the angular momentum or the orientation of the body. Since we desire to conserve momentum as well, we adjust the orientation. That is, the ordinary differential equation solver produces errors in kinetic energy and orientation, and our aim is to adjust those errors in orientation to conserve kinetic energy. We do this by analytically rotating the body to an orientation that has the same kinetic energy as the initial state. Since we do not change the angular momentum of the body, it is trivially conserved. Note that while this conserves angular momentum and energy, we do not properly resolve rotation, so the resulting orientations may still be inaccurate. If objects are moving very rapidly a large time step may cause collisions to be missed or resolved poorly. That said, the user is unlikely to notice that a collision occurred a frame too early or that the body had a slightly incorrect orientation when the collision normal was computed, nor is the user likely to notice that the body's orientation is inaccurate. Errors in momentum and energy, however, tend to produce noticeable artifacts.

2.2.1 Energy and Momentum Conserving Orientation

First, we show that there exist many nearby states that have the same kinetic energy and angular momentum. Consider the rotational kinetic energy of a body

$$KE = \frac{1}{2}\mathbf{L}^T\mathbf{I}^{-1}\mathbf{L},$$

where $\mathbf{L}$ is the angular momentum and $\mathbf{I}$ is the inertia tensor. If we change KE over time while keeping the angular momentum fixed, we have

$$KE' = \frac{1}{2}\mathbf{L}^T(\mathbf{I}^{-1})'\mathbf{L} = -\frac{1}{2}\mathbf{L}^T\mathbf{I}^{-1}\,\dot{\mathbf{I}}\,\mathbf{I}^{-1}\mathbf{L} \tag{2.1}$$

Using the orientation $\mathbf{R}$ and object space inertia tensor $\mathbf{I}_0$, where $\mathbf{I} = \mathbf{R}\mathbf{I}_0\mathbf{R}^T$, we can expand $\dot{\mathbf{I}}$ to

$$\dot{\mathbf{I}}=\dot{\mathbf{R}}\,\mathbf{I}_0\mathbf{R}^T + \mathbf{R}\mathbf{I}_0\,\dot{\mathbf{R}}^{\,T} =\dot{\mathbf{R}}\,\mathbf{R}^T\mathbf{I} + \mathbf{I}\mathbf{R}\,\dot{\mathbf{R}}^{\,T} = \mathbf{u}^*\mathbf{I} + (\mathbf{u}^*\mathbf{I})^T,$$

where we defined $\mathbf{u}^* =\dot{\mathbf{R}}\,\mathbf{R}^T$ as the cross product matrix for the vector $\mathbf{u}$ which represents our desired change in orientation. Substituting into (2.1),

$$\begin{aligned} KE' &= -\frac{1}{2}\mathbf{L}^T\mathbf{I}^{-1}(\mathbf{u}^*\mathbf{I} + (\mathbf{u}^*\mathbf{I})^T)\mathbf{I}^{-1}\mathbf{L} \\ &= -\mathbf{L}^T\mathbf{I}^{-1}(\mathbf{u}^*\mathbf{I})\mathbf{I}^{-1}\mathbf{L} = -\omega^T\mathbf{u}^*\mathbf{L} = -\mathbf{u}^T\mathbf{L}^*\omega. \end{aligned}$$

Since KE is to be conserved, $KE' = 0$ and therefore $\mathbf{u}^T\mathbf{L}^*\omega = 0$. Thus $\mathbf{u}$ must lie in the space spanned by $\mathbf{L}$ and ω, or

$$\mathbf{u} = c_1\omega + c_2\mathbf{L},$$

where c_1 and c_2 are constants. Note that in the special case when $\mathbf{L}$ and ω are parallel, $\mathbf{u}$ can actually be any vector. In this case there will be no errors in the trivial integration of orientation. However, in general we have a two-parameter family of equations that conserves both angular momentum and kinetic energy. This suggests that there are many nearby states (i.e. orientations) that have both the same kinetic energy and angular momentum.

2.2.2 Computing the Energy Fix

Let KE be the kinetic energy obtained after evolution

$$KE = \frac{1}{2}\mathbf{L}^T\mathbf{I}^{-1}\mathbf{L} = \frac{1}{2}\mathbf{L}^T\mathbf{R}\mathbf{I}_0^{-1}\mathbf{R}^T\mathbf{L},$$

and let KE_0 be the kinetic energy that should have been obtained by energy conservation. We would like to adjust the orientation obtained from evolution to fix the kinetic energy by replacing $\mathbf{R}$ with $e^{\mathbf{u}^*}\mathbf{R}$. Let $\mathbf{u} = \varepsilon\mathbf{s}$, with the unit vector $\mathbf{s}$ being the axis of rotation and ε being the amount to rotate. We will choose $\mathbf{s}$ such that it is orthogonal to $\mathbf{L}$. We set the kinetic energy of the modified post-evolution state equal to

$$KE_0 = \frac{1}{2}\mathbf{L}^T(e^{\mathbf{u}^*}\mathbf{R})\mathbf{I}_0^{-1}(e^{\mathbf{u}^*}\mathbf{R})^T\mathbf{L} = \frac{1}{2}((e^{\mathbf{u}^*})^T\mathbf{L})^T\mathbf{I}^{-1}((e^{\mathbf{u}^*})^T\mathbf{L}).$$

Expanding $e^{\mathbf{u}^*}$, where $\boldsymbol{\delta}$ is the identity matrix, we get

$$\begin{aligned} e^{\mathbf{u}^*} &= e^{\varepsilon\mathbf{s}^*} = \boldsymbol{\delta}\cos\varepsilon + \mathbf{s}^*\sin\varepsilon + \mathbf{s}\mathbf{s}^T(1-\cos\varepsilon) \\ (e^{\mathbf{u}^*})^T\mathbf{L} &= \mathbf{L}\cos\varepsilon - \mathbf{s}^*\mathbf{L}\sin\varepsilon + \mathbf{s}\mathbf{s}^T\mathbf{L}(1-\cos\varepsilon) \\ &= \mathbf{L}\cos\varepsilon - \mathbf{s}^*\mathbf{L}\sin\varepsilon. \end{aligned}$$

Substituting back into (2.2.2) yields

$$\begin{aligned} KE_0 &= \frac{1}{2}(\mathbf{L}\cos\varepsilon - \mathbf{s}^*\mathbf{L}\sin\varepsilon)^T\mathbf{I}^{-1}(\mathbf{L}\cos\varepsilon - \mathbf{s}^*\mathbf{L}\sin\varepsilon) \\ &= KE\cos^2\varepsilon - a\sin\varepsilon\cos\varepsilon + b\sin^2\varepsilon, \end{aligned} \tag{2.2}$$

where we have defined $a = (\mathbf{s}^*\mathbf{L})^T\omega$ and $b = \frac{1}{2}(\mathbf{s}^*\mathbf{L})^T\mathbf{I}^{-1}\mathbf{s}^*\mathbf{L}$. Note that ε is never used directly but instead appears only as $\sin\varepsilon$ and $\cos\varepsilon$. Letting $x = \cos\varepsilon$ and $y = \sin\varepsilon$,

$$KE_0 = KEx^2 - axy + y^2b. \tag{2.3}$$

Letting $c = KE - b$ and $k = KE - KE_0$, we obtain

$$\begin{aligned} axy &= k - cy^2 \\ a^2(1-y^2)y^2 &= (k-cy^2)^2 \\ 0 &= k^2 - (2kc + a^2)y^2 + r^2y^4, \end{aligned} \tag{2.4}$$

where $r = \sqrt{c^2 + a^2}$. Solving for y^2 we get

$$\begin{aligned} y^2 &= \frac{2kc + a^2 \pm \sqrt{(2kc+a^2)^2 - 4r^2k^2}}{2r^2} \\ &= \frac{1}{r^2}\left(kc + \frac{1}{2}a^2 \pm a\sqrt{\frac{1}{4}a^2 + k(c-k)}\right) \\ &= \frac{1}{2r^2}\Big(r^2 - cn \pm a\sqrt{r^2 - n^2}\Big), \end{aligned}$$

where $n = c - 2k$. Finally, if we let $q = a\sqrt{r^2 - n^2}$, then

$$y = \frac{p_s}{r}\sqrt{\frac{r^2 - cn + p_b q}{2}} \qquad x = \frac{p_c}{r}\sqrt{\frac{r^2 + cn - p_b q}{2}}, \tag{2.5}$$

where we have used p_s, p_c, and p_b (all equal to $\pm$ 1) to indicate signs that must be chosen.

From (2.3) we note that only the sign of xy, that is $p_s p_c$, is significant. For small angles of rotation, $\cos\varepsilon > 0$, so we choose $p_c = 1$. From (2.4) we choose $p_s = \mathrm{sgn}(k - cy^2)$. In the limit of very small errors in kinetic energy, the correction should also be very small, leading to the final sign choice $p_b = -\mathrm{sgn}(a)$.

2.2.3 Numerical Considerations

Direct use of (2.5) is not recommended. If $|k|$ is near floating point precision, then no significant error has been made and no correction is required. Similarly, if the kinetic energy is sufficiently small, then any energy increase that may have occurred is irrelevant to the simulation. Let $g = a^2 + 4k(c-k)$. If a/KE is very small (not

robust), or $g < 0$ (infeasible search direction), then a fallback search direction should be used. Let $p = a^2 + g + 2|a|\sqrt{g}$. Finally,

$$x = \frac{\sqrt{p + (c+n)^2}}{2r} \qquad \hat{y} = \frac{2|k|}{\sqrt{p + (c-n)^2}},$$

where $y = \text{sgn}(k - c\hat{y}^2)\hat{y}$.

2.2.4 Choosing a Rotation Axis

The previous section describes how to compute ε in a way that conserves the kinetic energy once a direction $\mathbf{s}$ is chosen. As discussed in section 2.2.1, if $\mathbf{u}$ is chosen as a linear combination of ω and $\mathbf{L}$, then kinetic energy is conserved. The locally optimal direction to choose is orthogonal to both, $\mathbf{s} = \mathbf{L}^*\omega / \|\mathbf{L}^*\omega\|$. This can be seen formally by noting that the Taylor series expansion of (2.2) is $KE_0 = KE - \mathbf{L}^T\mathbf{I}^{-1}\mathbf{s}^*\mathbf{L}\varepsilon = KE - \mathbf{s}^T(\mathbf{L}^*\omega)\varepsilon$. The magnitude of the linear term is optimized by choosing $\mathbf{s}$ to be in the direction of $\mathbf{L}^*\omega$.

One issue with choosing $\mathbf{s} = \mathbf{L}^*\omega$ is that only the angular momentum $\mathbf{L}$ remains constant as the object is rotated, and the angular velocity ω changes. Therefore, while $\mathbf{s} = \mathbf{L}^*\omega$ was a good choice for an instantaneous and small rotation, $\mathbf{L}^*\omega$ starts changing as the rotation is applied, and the best choice changes as well. Though locally optimal, this choice can fail for larger errors, in which case we simply choose a different $\mathbf{s}$.

At this point we fall back to using $\mathbf{s} = \mathbf{L}^*\mathbf{r}$, where $\mathbf{r}$ is either the maximum or minimum principal inertial direction, depending on the whether the current KE is greater than or less than KE_0. If $KE > KE_0$, then we want to rotate such as to align $\mathbf{L}$ and the largest principal inertial direction, $\mathbf{r}_{max}$. In this state, the body has the lowest KE it could possibly have for its angular momentum, so $\mathbf{s} = \mathbf{L}^*\mathbf{r}_{max}$. If $KE < KE_0$, then we want to align $\mathbf{L}$ and the smallest principal inertial direction, $\mathbf{r}_{min}$, and so $\mathbf{s} = \mathbf{L}^*\mathbf{r}_{min}$. Note that since we conserve angular momentum exactly and

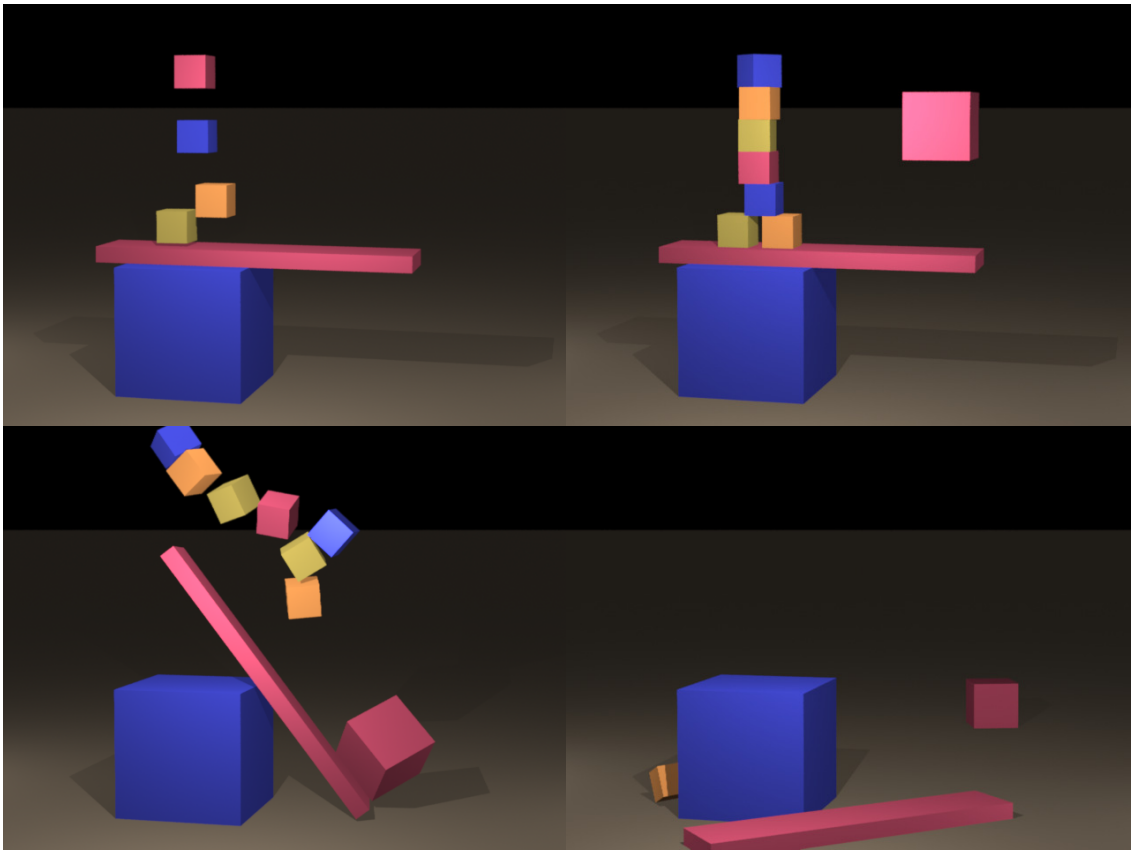


Figure 2.2: Collisions and contact behave correctly using our energy clamping. In this example, we drop several boxes to form a stack. A larger box then disrupts this stack and scatters the objects.

the maximum and minimum possible kinetic energy with this angular momentum is obtained by rotating in this way, this method of choosing directions is fail safe provided it is implemented robustly.

2.3 Collision and Contact Clamping

Contact and collision processing with Coulomb friction can also lead to energy gain, even in the otherwise very well-behaved example of a cube falling on the ground with a relatively small time step. Taking a large time step exacerbates this problem.

This limitation of the Coulomb friction model is also recognized and addressed in the literature, such as by replacing the Coulomb friction model with one that cannot increase energy [32]. However, this is insufficient in the case of collision or contact algorithms such as that of [64], since position and velocity live at different points in time when the collision impulse is computed and applied.

2.3.1 Clamping in Split States

Many time integration schemes include states where velocities and positions live at different times, a condition we refer to as a split state. Clamping energy while in a split state can lead to very serious problems. Consider the forward Euler scheme with gravity as the sole force, $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^n$ followed by $\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta t \mathbf{M}^{-1}\mathbf{g}$. Between the two updates lies a split state with position ahead of velocity. If the object is rising, so that $\mathbf{x}^{n+1} > \mathbf{x}^n$, one would calculate an increased potential energy and thus increased total energy in this split state. Clamping $\mathbf{x}^{n+1}$ to $\mathbf{x}^n$ removes this energy gain, but prevents the objects from rising under gravity. Similarly, if we consider the same forward Euler scheme computed in the reverse order, $\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta t \mathbf{M}^{-1}\mathbf{g}$ followed by $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^n$, we have a similar problem when falling under gravity. In the split state, we observe an increase in kinetic energy from gravity and clamp $\mathbf{v}^{n+1}$ to $\mathbf{v}^n$, preventing the object from falling correctly. From these examples, it is clear that conserving total energy is only meaningful when comparing non-split states, as momentary changes in total energy in split states are an inherent property of time integration schemes that utilize split states.

2.3.2 Clamping Impulses

In the case of both contact and collision processing, it is important to note that we do not modify positions. Therefore, potential energy is left unchanged, and preventing an increase in total energy is equivalent to preventing an increase in kinetic energy. First, we outline our general strategy for clamping in a split state. Then we explain

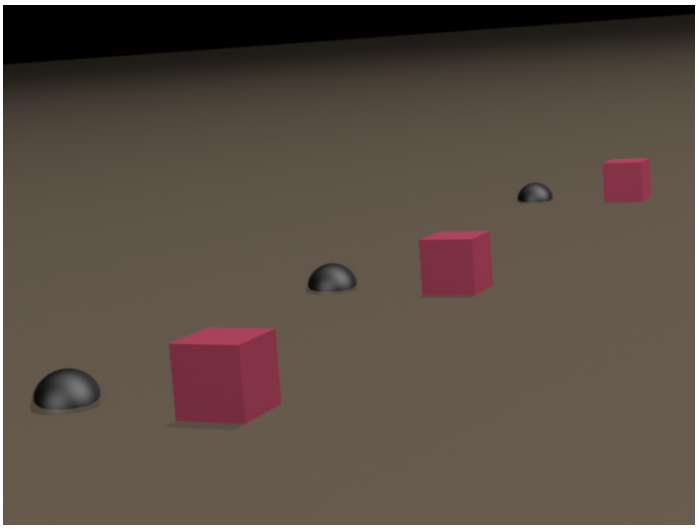


Figure 2.3: A rigid cube slides down an inclined plane and eventually stopping with friction, where the half sphere represents the analytic solution.

how it is applied to both collisions and contact.

Consider two bodies A and B. An impulse $\mathbf{j}$ and angular impulse $\mathbf{j}_\tau$ are to be applied to body A with the negations applied to body B to conserve momentum. We clamp the linear and angular impulses by scaling them by ε. Since kinetic energy is quadratic in velocity and the trivial solution $\varepsilon = 0$ results in an energy conserving impulse, there will always exist another real solution ε that can be found as follows. First, we write the kinetic energy of body A before and after the impulse is applied, KE_a and KE'_a respectively, in the non-split state.

$$KE_a = \frac{1}{2}\mathbf{p}_a^T m_a^{-1}\mathbf{p}_a + \frac{1}{2}\mathbf{L}_a^T \mathbf{I}_a^{-1}\mathbf{L}_a$$
$$KE'_a = \frac{1}{2}(\mathbf{p}_a + \varepsilon\mathbf{j})^T m_a^{-1}(\mathbf{p}_a + \varepsilon\mathbf{j}) + \frac{1}{2}(\mathbf{L}_a + \varepsilon\mathbf{j}_{\tau a})^T \mathbf{I}_a^{-1}(\mathbf{L}_a + \varepsilon\mathbf{j}_{\tau a}),$$

where $\mathbf{p}$ is the linear momentum, m is the mass, and $\mathbf{j}_{\tau a} = \mathbf{r}_a^* \mathbf{j} + \mathbf{j}_\tau$. Note that both $\mathbf{j}$ and $\mathbf{j}_\tau$ will be computed in a split state, and in particular $\mathbf{r}_a$ will also be computed in that split state. This simply means that the impulses used to compute KE'_a will come from a split state computation, but as long as equal and opposite impulses are applied at the same point in space we will conserve linear and angular momentum.

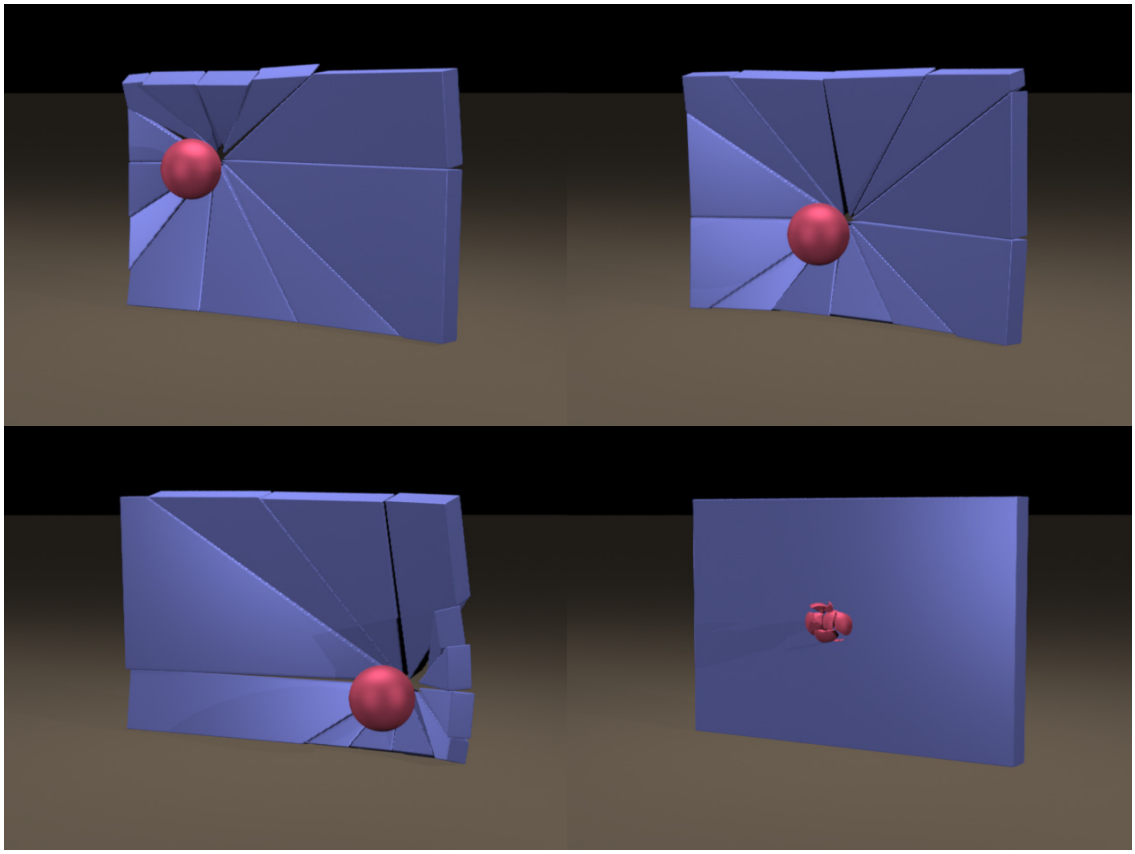


Figure 2.4: Our algorithm for prescoring all of space allows us to center a fracture pattern around the point of impact. The above series of images shows a wall being fractured at 3 separate locations, with all the fractures being generated from the same fracture pattern. Finally, a sphere fractures as it hits the wall using the same fracture pattern. The level set resolution of all the walls used in our examples is $150 \times 10 \times 100$.

That is, split state impulses have no effect on conservation of momentum and are fairly standard in the literature. However in our kinetic energy calculation, it is important to view every term in KE_a and KE'_a as existing at a non-split state.

The change in the kinetic energy of body A is given by

$$KE'_a - KE_a = \varepsilon(\mathbf{j}^T m_a^{-1}\mathbf{p}_a + \mathbf{j}_{\tau a}^T \mathbf{I}_a^{-1}\mathbf{L}_a) + \frac{1}{2}\varepsilon^2(\mathbf{j}^T m_a^{-1}\mathbf{j} + \mathbf{j}_{\tau a}^T \mathbf{I}_a^{-1}\mathbf{j}_{\tau a}),$$

and the total change in kinetic energy including both bodies is

$$\begin{aligned} KE' - KE = &\varepsilon(\mathbf{j}^T m_a^{-1}\mathbf{p}_a + \mathbf{j}_{\tau a}^T \mathbf{I}_a^{-1}\mathbf{L}_a - \mathbf{j}^T m_b^{-1}\mathbf{p}_b - \mathbf{j}_{\tau b}^T \mathbf{I}_b^{-1}\mathbf{L}_b) \\ &+ \frac{1}{2}\varepsilon^2(\mathbf{j}^T m_a^{-1}\mathbf{j} + \mathbf{j}_{\tau a}^T \mathbf{I}_a^{-1}\mathbf{j}_{\tau a} + \mathbf{j}^T m_b^{-1}\mathbf{j} + \mathbf{j}_{\tau b}^T \mathbf{I}_b^{-1}\mathbf{j}_{\tau b}). \end{aligned}$$

Setting $KE' - KE = 0$ to preserve the kinetic energy results in the trivial solution of $\varepsilon = 0$ as well as the nontrivial solution

$$\varepsilon = \frac{2(\mathbf{j}^T m_b^{-1}\mathbf{p}_b + \mathbf{j}_{\tau b}^T \mathbf{I}_b^{-1}\mathbf{L}_b - \mathbf{j}^T m_a^{-1}\mathbf{p}_a - \mathbf{j}_{\tau a}^T \mathbf{I}_a^{-1}\mathbf{L}_a)}{\mathbf{j}^T m_b^{-1}\mathbf{j} + \mathbf{j}_{\tau b}^T \mathbf{I}_b^{-1}\mathbf{j}_{\tau b} + \mathbf{j}^T m_a^{-1}\mathbf{j} + \mathbf{j}_{\tau a}^T \mathbf{I}_a^{-1}\mathbf{j}_{\tau a}}.$$

If $\varepsilon > 1$, then the impulse required to conserve kinetic energy is larger than the impulse we are applying. This simply means that Coulomb friction is reducing the kinetic energy and nothing need be done. If $0 < \varepsilon < 1$, the proposed impulse is erroneously increasing the kinetic energy and thus we scale the impulses by epsilon in order to prevent any energy increase. Note that $\varepsilon < 0$ is impossible in a non-split state but can occur in a split state, representing an infeasible direction. In this case, we choose $\varepsilon = 0$, meaning that we discard the impulse altogether. Note that discarding the impulse leads to loss of friction, but in practice this case is very rare.

When one of the two bodies has infinite mass, such as the static ground plane or a moving kinematic body (consider for example an elevator), the above equations need a slight modification. For the case of a static body, all the terms pertaining to the body vanish as m^{-1} and $\mathbf{I}^{-1}$ are zero. One might assume the same applies to a kinematic body with non-trivial velocity, however this would preclude an object from being lifted with an elevator. In particular, the terms relating to the infinite mass body only cancel in the sense of relative velocities. Thus, for a collision with a kinematic object we rewrite the velocities of the dynamic body in the relative frame of the kinematic object based on its pointwise velocity at the point of collision. Then the clamping can be done computing ε in the same manner as for a static body. This preserves kinetic energy in the moving reference frame, giving the desired effect. Note that infinitely massive bodies may transfer kinetic energy to other bodies without change in velocity.

Collision Clamping The collision algorithm of [64] applies collisions to the split state $\mathbf{x}^{n+1}$, $\mathbf{v}^n$. However, we have a valid non-split state to clamp against, since we can clamp the impulses applied to $\mathbf{v}^n$ in the state $\mathbf{x}^n$, $\mathbf{v}^n$. This means that KE is computed using entirely time n quantities, as are all terms in KE' except those relating to $\mathbf{j}$ and $\mathbf{j}_\tau$. Instead, $\mathbf{j}$ and $\mathbf{j}_\tau$ are computed using the split state, just as in [64] where in particular $\mathbf{r}_a$ (and $\mathbf{j}_{\tau a}$) are also computed using that split state.

Contact Clamping The contact algorithm of [64] also involves a split state. We use a Newmark variant of this scheme introduced by [134], which performs two slightly different contact steps, the first as part of a position update and the second as part of a velocity update, rather than one as in [64]. The second contact step involves the split state $\mathbf{x}^{n+2}$, $\mathbf{v}^{n+1}$, but we observe that we can clamp against the non-split state $\mathbf{x}^{n+1}$, $\mathbf{v}^{n+1}$ using the method described above. This is identical to the treatment of collisions, where the split state was formed by updating the position one time step while leaving the velocity the same.

The first contact step involves the split state $\mathbf{x}^{n+1}$, $\mathbf{v}^{n+1/2}$. Since the only available non-split state at that point in the integration is at $\mathbf{x}^n$, $\mathbf{v}^n$, we tried clamping there. Unfortunately, clamping against that state leads to problems. For example, if bodies are initially at rest and in contact, any impulse will change $\mathbf{v}^n$ away from zero and increase energy, resulting in the impulse being clamped to zero. This prevents contacting bodies at rest from applying contact impulses and leads to jittering. Therefore, we do not clamp during the first contact step, and we have not observed any ill effects from this omission. This may be partially due to the fact that this first contact step is only used to update the position, and its effects on the velocity are removed. Therefore, energy gain here may be less detrimental than during the second contact step or during the collision processing where the results are used to directly modify the velocity. However, in spite of our ability to obtain stable and accurate simulations at the frame rate, we do think that clamping the first contact step should be addressed as future work. That is, while the energy behavior of the scheme is highly improved by limiting any kinetic energy growth during evolution,

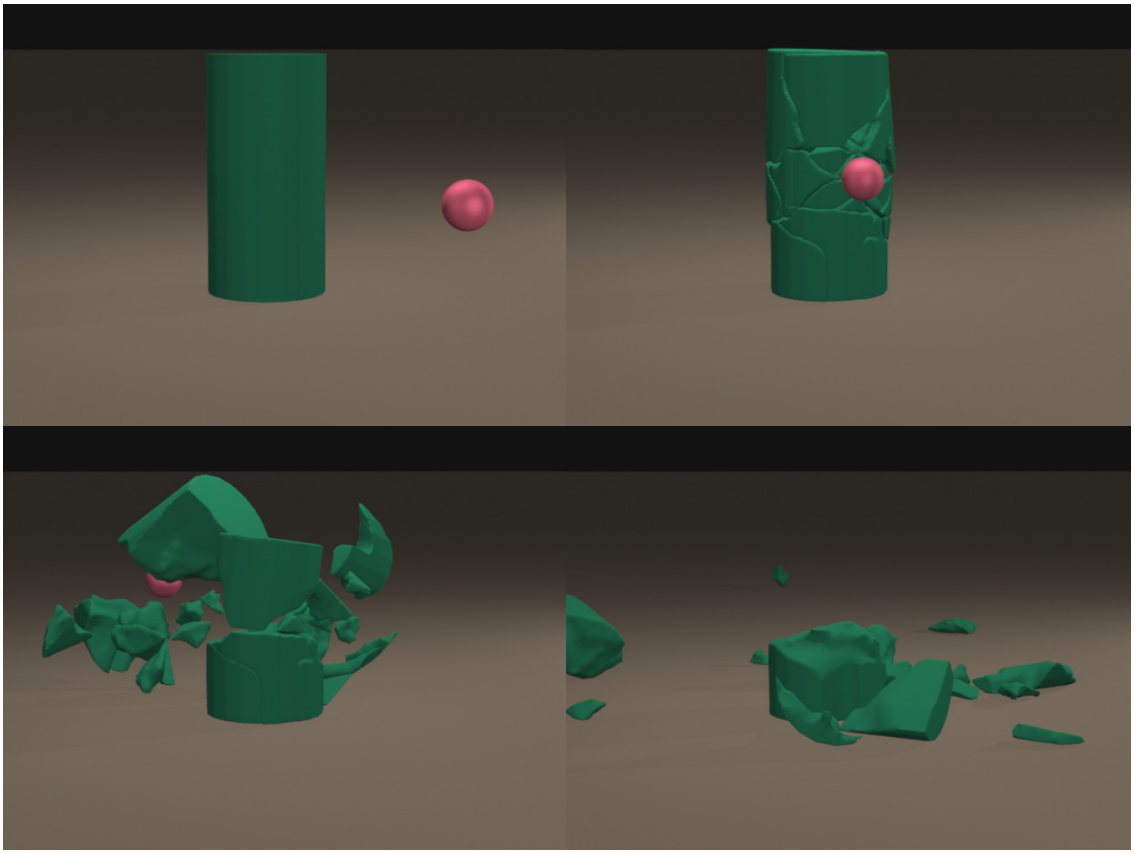


Figure 2.5: A cylinder (level set resolution of $55 \times 105 \times 55$) is shattered into 25 pieces by a sphere. This simulation averaged 22 frames/sec on a single 3.00GHz Xeon core using only algorithmic optimizations.

collisions, and the contact phase of the velocity update, there may be scenarios where the kinetic energy growth during the contact phase of the position update could have adverse effects.

Observe that this approach to clamping does not rely on any of the details of the collision or contact algorithms being employed, beyond the relatively simple assumption that it will result in an impulse to be applied. It can be applied when the state is non-split, or when the state is split but a suitable non-split state is available. It is equally suitable for clamping the impulses obtained from any other process that affects the state only through the impulse computed.

2.4 Fracture

Fracture of rigid bodies has typically been achieved through prescoring methods or by dynamically computing the stresses of the object to determine how to fracture it. Prescoring is fast but constrains the artist's control over the look of the fracture. The artist could make a realistic fracture pattern emanating from a particular point on the object and hope that the point of impact is close to the fracture center or make a somewhat uniform fracture pattern that is not centered about any point and is not very realistic. Computing fracture dynamically from the internal stresses of an object produces realistic, point-of-impact-centered patterns but requires significantly more computation time and is therefore impractical for real-time applications. The goal of our method is to combine the speed of a prescoring technique with the point-of-impact-centered realism of stress-computing methods. We achieve this by *prescoring all of space*, generating a fracture pattern that emanates from a fracture center, which can then be aligned on-the-fly with the collision location on the object we wish to fracture. This approach provides artists with the ability to more precisely control how an object breaks around a collision point while still being feasible for real-time applications.

2.4.1 Framework

Rigid Body Representation As we will be implementing our fracture algorithm in the context of [64], it is important to note that for their implementation, they represented rigid bodies with a triangulated surface mesh and a volumetric level set. Contact and collisions are both detected and processed by computing level set data from one object at the positions of the surface particles of the other object. We note that these algorithms do not require surface triangles but rather only the vertices. Thus, each of our rigid bodies will contain a position, orientation, velocity, angular velocity, mass, inertia tensor, list of surface particles, and volumetric level set function. Any standard fast dual-contouring [85] or marching cubes method [100] can be used

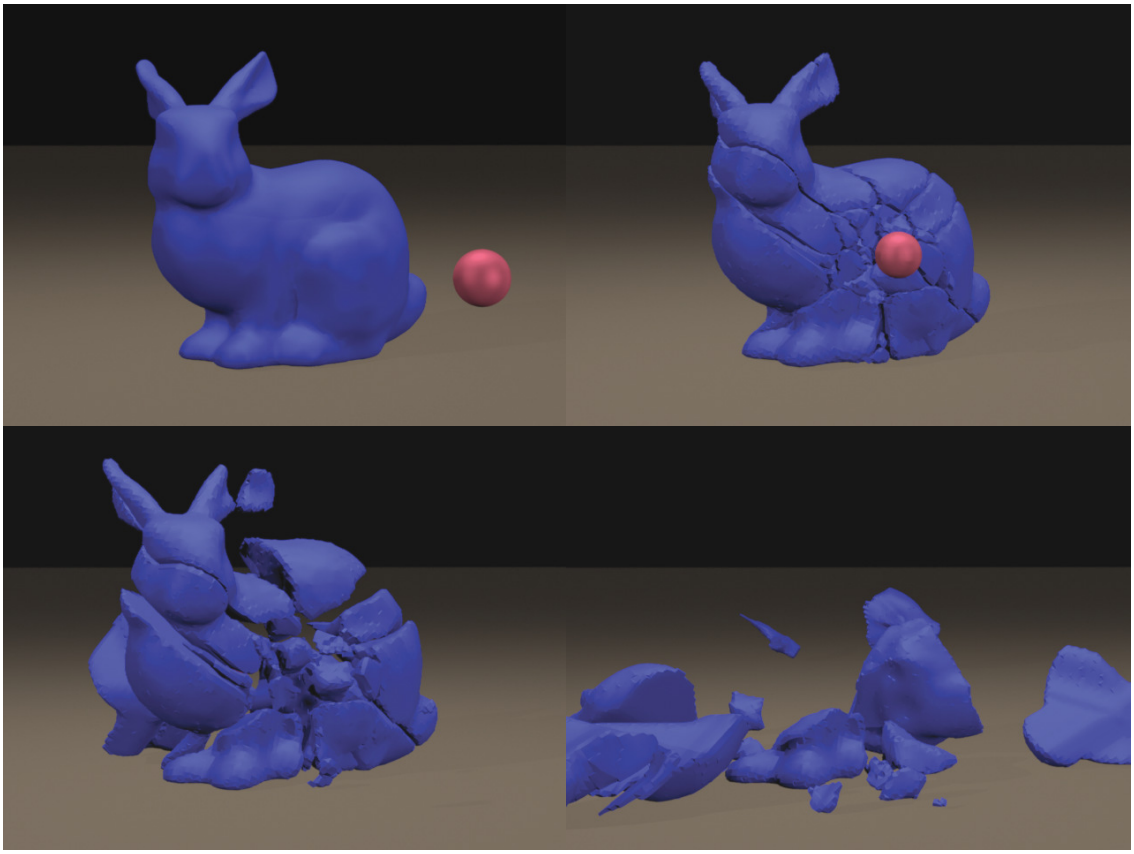


Figure 2.6: A complex bunny model with a level set resolution of $70 \times 60 \times 45$ is shattered into 35 pieces.

to triangulate the level set for rendering.

Fracture Patterns The goal is to fracture all of space, and this is easily accomplished by fracturing any piece of geometry that can subsequently be translated and rescaled to enclose the dynamically simulated object to be fractured. For our examples we simply used the bounding box of the object and fractured this box into a number of regions. A single point in this cube is defined as the point of impact for the fracture, and a coordinate system is attached to that point to represent the collision normal. When an object fractures, this box will be translated and rotated to align with the collision and scaled to enclose the object, both important for the artist

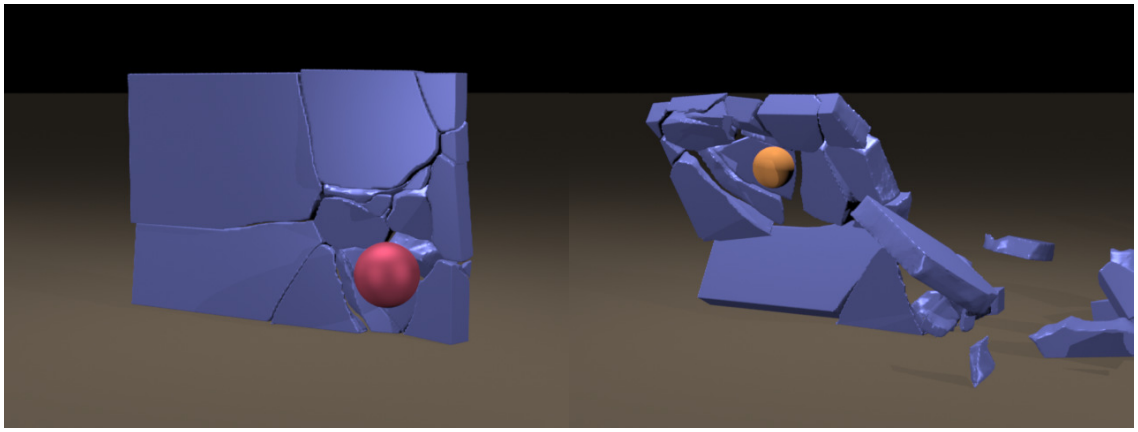


Figure 2.7: A wall is fractured in the lower right corner with a sphere, creating large pieces away from the fracture location (left). Then, a second sphere fractures one of the large fragments (right).

to keep in mind when sculpting the pieces. Each of the fracture regions represents a fragment of the fractured space and stores a list of surface particles, a volumetric level set function, and the location of the fracture center with respect to the region.

2.4.2 Fracturing a Rigid Body

Each rigid body in our simulation is assigned a fracture threshold, which represents the magnitude of the collision impulse the body must feel before it will fracture. When a pair of rigid bodies is processed for collision, we compute the impulse that would be applied due to the collision, and if it is above the fracture threshold of the body, we proceed to fracture the body. The result of this fracture will be a set of fragment bodies, each of which is the result of intersecting the original body, which we refer to as the parent body, with one of the regions of the fracture pattern.

Level Set To generate the level set of a fragment, we will merge the level sets of the fracture region and parent body that intersect to form the fragment. The level set of the fragment is computed at each point as the maximum of the fracture region's and parent body's level set values at that point. This algorithm will produce a level

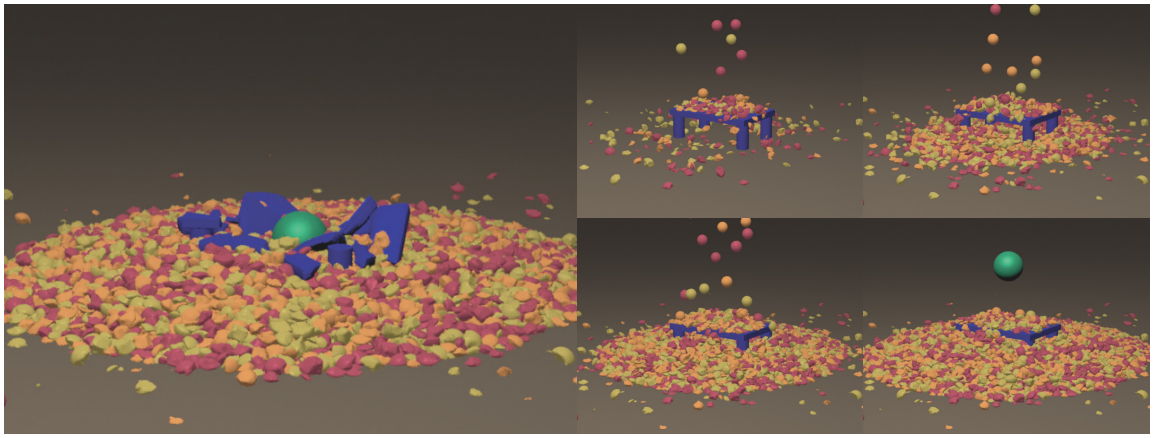


Figure 2.8: 1000 small spheres are dropped onto a table. Then, a larger sphere is dropped to fracture the table, resulting in 8992 rigid body fragments. The small spheres have a level set resolution of $55 \times 55 \times 55$, and the table top has a resolution of $150 \times 10 \times 100$.

set function that represents the correct signed distance value within the fragment body but may underestimate the signed distance outside the fragment body. This does not cause problems, since we still correctly answer inside/outside queries, and such queries are the only usage of level set values *outside* a body by the collision and contact algorithms of [64]. The level set of the intersection could be disconnected, so we flood fill the level set, giving each connected component a unique color and a separate fragment.

Besides this basic algorithm, we propose an aggressive optimization. If one disregards the notion of a collision normal when aligning the prescored fracture pattern and the parent body and allows up to a half-grid-cell perturbation of the point of collision when aligning the parent body and the fracture pattern, it is possible to overlay the prescored level set and the parent body's level set in the material space of the parent body. This means that one can avoid all interpolation when computing level set values and simply visit each grid point, comparing the two level set values that exist there. We use this optimization in all of our examples.

Surface Particles When constructing a new dynamic fragment of the subsequently fractured parent body, one must find all the particles on the fragment's surface. Those particles come from the surfaces of the fracture region in question and the parent body we are fracturing, and we just need to determine which particles are retained and which particles are discarded. The most straightforward way to test a particle from one object against the other object is to compute its level set value on the other object and see if it inside or outside that object. Particles outside the other object are discarded, and those inside the other object are will help form the new surface and are retained. After doing this for each object against the other, one obtains the final set of all surface particles. However if a single fracture region results in two or more separate disconnected pieces when intersected with the parent body, then we have not properly divided the particles between these pieces. This is easily remedied during the flood fill stage that identifies these connected components as the particles lie closer to the zero contour of one level set than the other, and they are simply assigned to the fragment to which they are closer, i.e. the one that has a smaller level set value at the location.

When using the optimization mentioned above that was used to exactly align a fracture pattern's level set with that of the parent body's in material space, a further optimization can be done for assigning particles to each region. One can simply assign particles to voxels, both for the parent rigid body and for each fracture region. Then when the flood fill process determines which fragment a voxel belongs to, the particles contained in that voxel are simply assigned to it.

Inertial Properties Once we have the level set and particles for a fragment, we need to compute the inertial properties of the fragment. Since we do not store a surface mesh for the fragments, we instead observe that we can compute the desired inertial properties from the level set. The volume of the parent V_p can be computed from its level set region $\mathbb{P}$ as $V_p = \int_{\mathbb{P}} d\mathbf{x}$. The density of the parent is given by $\rho_p = m_p/V_p$. The density of a subsequent fragment ρ_f is set equal to that of the parent. Then, we compute a few more quantities from the fragment's level set region

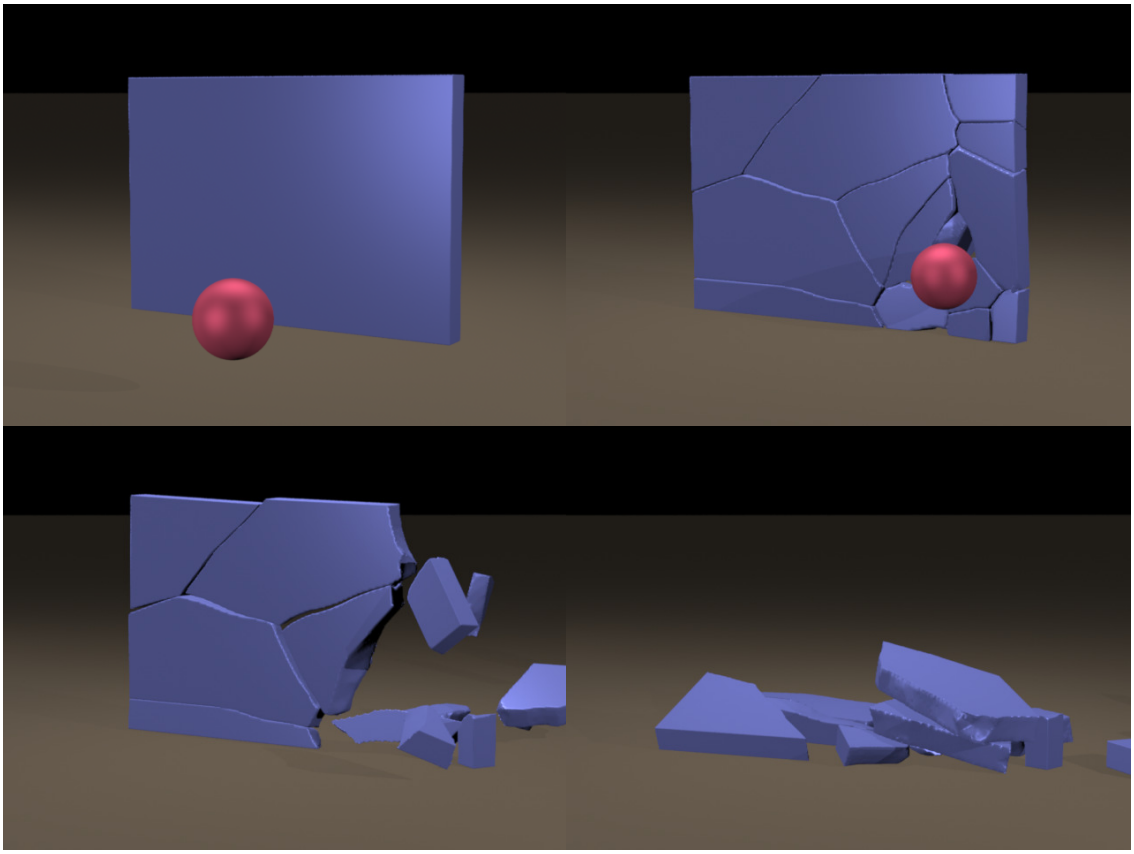


Figure 2.9: A wall is fractured using a realistic fracture pattern.

$\mathbb{F}$,

$$V_f = \int_{\mathbb{F}} d\mathbf{x}, \quad \mathbf{C}_f = V_f^{-1} \int_{\mathbb{F}} \mathbf{x}\, d\mathbf{x}, \quad \mathbf{\Sigma}_f = \int_{\mathbb{F}} \mathbf{x}\mathbf{x}^T\, d\mathbf{x},$$

where V_f and $\mathbf{C}_f$ are the volume and center of mass of the fragment. Finally, the mass is $m_f = \rho_f V_f$, and the inertia tensor is $\mathbf{I}_f = \rho_f(\mathbf{tr}(\mathbf{\Sigma}_f)\boldsymbol{\delta} - \mathbf{\Sigma}_f)$. We can simplify these computations by treating a level set cell as entirely inside or outside the fragment based on the sign of its center.

Position and Orientation We choose the position of the fragment to be its center of mass, and we diagonalize the inertia tensor to obtain the orientation of the fragment. We are careful to transform the surface particles of the fragment into the new coordinate system. We also store this same transform along with the level set.

Velocity and Angular Velocity We compute velocity and angular velocity to conserve both momentum and angular momentum, as in [104]. That is, $\omega_f = \omega_p$ and $\mathbf{v}_f = \mathbf{v}_p + \omega_p \times \mathbf{r}_f$, where $\mathbf{r}_f$ points from the center of mass of the parent to the center of mass of the fragment.

2.4.3 Generating Fracture Patterns

We do not propose any particular method for fracture pattern generation, though we explore a few possibilities for doing so. The most straightforward is for an artist to design the fracture regions in the form of surface meshes for those regions using existing tools in much the same way that they prescore models today; the main difference is that the artist would prescore a large cube rather than prescoring individual models. Alternatively, these surfaces may be computed using offline finite element analysis in which some portion of space is fractured. From the surface meshes, the level sets can be calculated using a standard algorithm such as the fast marching method. The surface particles might be the vertices of the surface mesh, vertices of a refined surface mesh, or even simply particles sampled on the mesh depending on the quality and refinement of the sculpted mesh.

2.4.4 Alternatives to Level Sets

We chose to use level sets for our fracturing implementation because we need level sets for our collision algorithms already, and implementing the proposed fracturing algorithm is particularly straightforward to do using level sets. It should be noted, however, that level sets have a number of limitations. Level sets tend to be memory intensive, and it is difficult to properly resolve thin geometry or shards at a reasonably coarse resolution. Due to the level set based collisions, the fractured pieces do not fit snugly together after fracture, but since fracturing tends to be energetic, we have not found this to be a problem. The same high level fracturing algorithm can be performed using only surface meshes by performing Boolean operations of the meshes against

the fracture patterns. We note that while such Boolean operations are complicated to implement robustly, implementations are available. In this case, the inertial properties of the fragments should be computed from their surface meshes.

2.5 Examples

We ran all of our simulations on a single 3.00GHz Xeon core using only the algorithmic optimizations proposed in this paper, i.e. no aggressive hardcoding, special libraries, or other hardware tricks were used. In Figure 2.4, we show a sphere fracturing a wall at different locations, all using a single radial fracture pattern. We then lower the fracture threshold of the ball and increase the threshold of the wall, allowing the ball to shatter while the wall does not. These examples show that our method can use a single fracture pattern to generate point-of-impact-centered fracture of any object in the scene. Figure 2.9 shows the same scenario, but with a more realistic fracture pattern. All of these simulations ran at 15 frames/second or faster.

In Figure 2.5, we show that we can handle the fracture of other objects, simulating a sphere fracturing a tall cylinder at 22 frames/second. Figure 2.6 illustrates that we can handle more complicated geometry, simulating a sphere fracturing a bunny into 35 fragments at 4 frames/second. Figure 2.7 illustrates that the fragments produced by our fracture algorithm are just normal rigid bodies and can themselves be fractured.

Finally, Figure 2.8 shows that our algorithms scale to a large number of rigid bodies, shattering 1000 spheres on a table top and finally shattering the table itself. After the table top fractures, there are 8992 rigid bodies in the scene. We also ran the same example with 25 and 100 spheres. For the 25 sphere drop, the early frames averaged 25 frames/second while the later frames averaged 3 frames/second. For the 100 sphere drop, the early frames averaged 25 frames/second while the later frames averaged .26 frames/second. For the 1000 sphere drop, the early frames averaged 25 frames/second while the later frames averaged 5 minutes/frame with 8992 fragment bodies in contact on the ground.

2.6 Conclusion

We present a method for conserving both kinetic energy and angular momentum during rigid body evolution as well as a method for clamping kinetic energy during contact and collision processing. These methods allow for stable rigid body simulations at the frame rate. Moreover, we introduce a method for *prescoring all of space*, allowing realistic point-of-impact-centered fracture of rigid bodies that runs efficiently on modern machines.

Though our methods for mitigating energy gain allow us to achieve stable simulations while taking time steps at the frame rate, there may still be situations in which energy can increase, as we do not clamp kinetic energy during the contact phase of the position update. Although we did not notice any visual artifacts, we believe that this should be addressed as future work.

Chapter 3

Incompressible Solids

In this chapter, the focus switches from rigid bodies to deformable bodies. The Poisson's ratio of a material provides an indication of how resistant an elastic material is to changes in its volume. When a block of material is compressed in one direction, it will normally expand in the other two (transverse) directions. The ratio of the expansion in a transverse direction divided by the compression applied is the Poisson's ratio. If the material conserves its volume exactly, the Poisson's ratio will be 0.5. Many constitutive models, such as the neo-Hookean model, break down under this limit. This manifests numerical in the form of locking, where the material is excessively and non-physically resistant to any changes in shape. This chapter addresses a method of simulating incompressible materials by treating the incompressibility of a material as a constraint separate from its constitutive properties. This constraint is enforced using a pressure projection as is commonly done when simulating incompressible fluids. The material in this chapter is based on the published work of [80].

3.1 Introduction

Recently virtual humans have received increased attention for modeling stunt doubles, virtual surgery, etc. When modeling virtual humans, one needs to consider shape changes dictated by muscles, skin, fat, and other organs. These soft biological tissues are highly incompressible and involve complicated constitutive models including anisotropy and both active and passive components. Notably, volume in biological tissues is conserved locally, and it is insufficient to only conserve the total volume. Besides realistic modeling of tissues for virtual humans, volume preservation is important in its own right. [95] states, "The most important rule to squash and stretch is that, no matter how squashed or stretched out a particular object gets, its volume remains constant."

Although our interest is in physically based simulation, constant volume deformations are also of interest in shape modeling, e.g. [174, 2, 163]. Several authors have proposed methods that conserve total but not local volume, e.g. [121, 122], and [74] proposed an ad hoc method to address the "undesirable behaviors" caused by conserving only total volume.

A number of authors have considered approximate local volume preservation using simple spring-like forces, e.g. [39, 106, 25, 102, 152]. In the area of finite element simulation, [118] added a volume preserving force to each tetrahedron, a technique similar to the notion of quasi-incompressibility (see [139]) which has been used extensively in finite element simulations of muscle tissue [170, 151]. See also [120, 42]. The problem with mass-spring and quasi-incompressible formulations is that they only provide a force towards volume preservation, and therefore volume is not preserved in the presence of competing forces. This can be alleviated to some extent by increasing the stiffness of the volume-preserving forces, but this competes with and can overwhelm the other forces in the model.

In fluid dynamics, volume preservation is addressed by decomposing a vector field into the gradient of the pressure plus a divergence-free part and subsequently discarding

the gradient (see e.g. [52]). By introducing this pressure variable, one discards compressible motions while retaining those orthogonal to volume change. [109] proposed a fluid dynamics approach to incompressible deformable solids, but used artificial compressibility (requiring ad hoc volume adjustments) rather than fully divergence-free velocities and did not consider constitutive models or elastic forces in the object's interior. We take a fluid dynamics approach to deformable solids as well, introducing a pressure variable into a standard finite volume approximation. Others, such as [127], have used an independent pressure variable for similar purposes. Besides using the pressure to obtain a divergence-free velocity, we also project the positions to exactly conserve volume avoiding error accumulation (note that Eulerian fluids do not have a position variable). Similar approaches are currently receiving attention in the computational mechanics literature, see e.g. [43, 112, 92, 19, 20, 126, 38]. In contrast to most of these works, we present a simple technique independent of any particular constitutive model or time integration scheme so that it is easily integrated into any finite element solver. Moreover, we show how to integrate this incompressibility constraint with other possibly competing constraints, in particular object contact and self-contact.

3.2 Time Discretization

Regardless of the time integration scheme, our goal is to make the velocity divergence free as well as to update the positions in a manner that moves the nodes to maintain constant volume in one-rings. When processing collisions or treating volume errors, some methods can only modify the position via adjustments to the velocity. Unfortunately, this requires $O(1)$ velocity changes to make $O(\Delta x)$ changes in position (since $\Delta t \sim \Delta x$). Thus, we propose a method that treats errors in position and velocity separately, ensuring that volume errors due to collisions or other phenomena can be corrected without introducing oscillations. Note that structural integration methods such as [92, 89] do not have this property: in order to quickly correct $O(\Delta x)$ errors in volume they must produce $O(1)$ velocities.

For concreteness, we give the particular time integration scheme used for our examples below, with the two additional steps required for incompressibility highlighted, and then explain these new steps in detail. We used a modified version of the semi-implicit Newmark scheme of [28] (see [130] for details). A step of size Δt from $(\mathbf{x}^n, \mathbf{v}^n)$ to $(\mathbf{x}^{n+1}, \mathbf{v}^{n+1})$ proceeds as follows:

1. $\mathbf{v}_\star^{n+1/2} = \mathbf{v}^n + \frac{\Delta t}{2}\mathbf{a}(t^{n+1/2}, \mathbf{x}^n, \mathbf{v}_\star^{n+1/2})$
2. $\tilde{\mathbf{v}}^{n+1/2} = \mathbf{v}_\star^{n+1/2} + \boldsymbol{\gamma}_x$ **(to correct positions)**
3. Modify $\tilde{\mathbf{v}}^{n+1/2}$ with elastic and inelastic self-repulsions
4. $\tilde{\mathbf{x}}^{n+1} = \mathbf{x}^n + \Delta t\tilde{\mathbf{v}}^{n+1/2}$
5. Collide with objects to obtain $\mathbf{x}^{n+1}$ and $\mathbf{v}_\star^n$
6. $\tilde{\mathbf{v}}^n = \mathbf{v}_\star^n + \boldsymbol{\gamma}_v$ **(to correct velocities)**
7. $\mathbf{v}^{n+1/2} = \tilde{\mathbf{v}}^n + \frac{\Delta t}{2}\mathbf{a}(t^{n+1/2}, \mathbf{x}^{n+1/2}, \mathbf{v}^{n+1/2})$
8. $\mathbf{v}^{n+1} = 2\mathbf{v}^{n+1/2} - \tilde{\mathbf{v}}^n$
9. Modify $\mathbf{v}^{n+1}$ for inelastic self-repulsions and friction

where $\mathbf{x}^{n+1/2} = (\mathbf{x}^n + \mathbf{x}^{n+1})/2$ in step 7 is the average of the initial and final positions and $\mathbf{a}(t, x, v)$ is the acceleration due to all forces except collisions and incompressibility. Step 1 is a backward Euler solve to obtain a velocity for use in the position update, and we adjust this velocity in step 2 so that step 4 corrects the volume in each one-ring. After colliding with kinematic objects in step 5, steps 7 and 8 advance the velocity field forward in time, and we apply a correction in step 6 to make the velocity field divergence free before time evolution.

If we temporarily ignore collisions, steps 2 and 4 combine to form

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t\mathbf{v}_\star^{n+1/2} + \Delta t\boldsymbol{\gamma}_x.$$

This formula is valid for any time integration scheme that computes $\mathbf{x}^{n+1}$ from $\mathbf{x}^n$ by

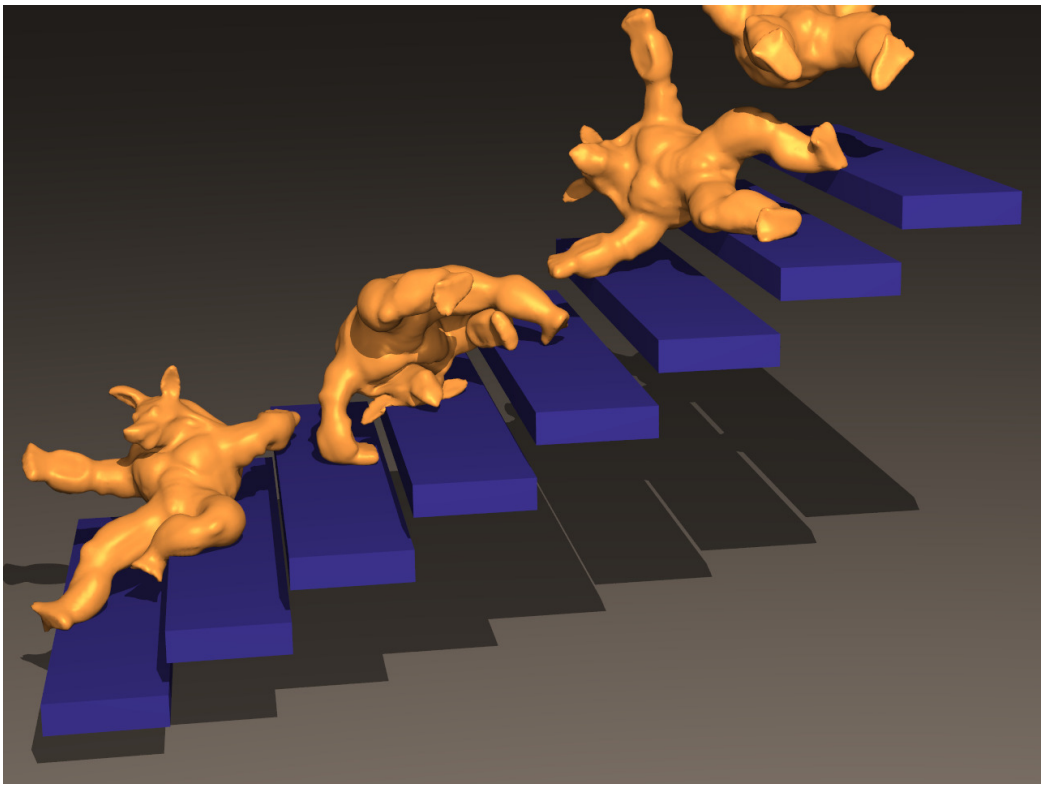


Figure 3.1: An incompressible elastic armadillo falls down a flight of stairs preserving its volume to an accuracy of 0.1%.

defining $\mathbf{v}_\star^{n+1/2} = (\mathbf{x}^{n+1} - \mathbf{x}^n)/\Delta t$. Since volume is to be conserved, the final volumes should equal the initial volumes, i.e. $\mathbf{V}(\mathbf{x}^{n+1}) = \mathbf{V}(\mathbf{x}^0)$. Substituting for $\mathbf{x}^{n+1}$ and linearizing gives

$$\mathbf{V}(\mathbf{x}^0) = \mathbf{V}(\mathbf{x}^{n+1}) = \mathbf{V}(\mathbf{x}^n + \Delta t \mathbf{v}_\star^{n+1/2} + \Delta t \boldsymbol{\gamma}_x) \approx \mathbf{V}(\mathbf{x}^n) - \Delta t \mathbf{G}^T(\mathbf{v}_\star^{n+1/2} + \boldsymbol{\gamma}_x),$$

where $-\mathbf{G}^T$ is the volume-weighted divergence (see Section 3.3). Similar to the typical pressure correction in fluids, we use

$$\tilde{\mathbf{v}}^{n+1/2} = \mathbf{v}_\star^{n+1/2} - \mathbf{M}^{-1}\mathbf{G}\hat{\mathbf{p}} \qquad \boldsymbol{\gamma}_x = -\mathbf{M}^{-1}\mathbf{G}\hat{\mathbf{p}},$$

where $\hat{p} = \Delta t p$ is the scaled pressure, $\mathbf{G}$ is the volume-weighted gradient (see Section 3.3), and $\mathbf{M}$ is the diagonal mass matrix. Substituting into (3.2) we have

$$\mathbf{V}(\mathbf{x}^n) - \Delta t \mathbf{G}^T \mathbf{v}_\star^{n+1/2} + \Delta t \mathbf{G}^T \mathbf{M}^{-1} \mathbf{G} \hat{\mathbf{p}} = \mathbf{V}(\mathbf{x}^0).$$

Rearranging into standard Poisson equation form,

$$\mathbf{G}^T \mathbf{M}^{-1} \mathbf{G} \hat{\mathbf{p}} = \mathbf{G}^T \mathbf{v}_\star^{n+1/2} - \frac{1}{\Delta t}(\mathbf{V}(\mathbf{x}^n) - \mathbf{V}(\mathbf{x}^0)), \tag{3.1}$$

which can be solved for $\hat{\mathbf{p}}$ and then $\boldsymbol{\gamma}_x = -\mathbf{M}^{-1}\mathbf{G}\hat{\mathbf{p}}$. Note that $\mathbf{M}$ is $3n \times 3n$, $\mathbf{G}$ is $3n \times n$, $\mathbf{v}_\star^{n+1/2}$ is a $3n$-vector, and $\hat{\mathbf{p}}$, $\mathbf{V}(\mathbf{x}^n)$, and $\mathbf{V}(\mathbf{x}^0)$ are n-vectors. Finally, we stress that (3.1) corresponds to a single step of Newton's method applied to the equation $\mathbf{V}(\mathbf{x}^{n+1}) = \mathbf{V}(\mathbf{x}^0)$.

We correct the velocity to be divergence free in step 6, although this can be executed at any point in the algorithm since it is a static projection. Taking the divergence of step 6 and setting $-\mathbf{G}^T \tilde{\mathbf{v}}^n = 0$ yields $0 = -\mathbf{G}^T \mathbf{v}_\star^n + -\mathbf{G}^T \boldsymbol{\gamma}_v$, where $\boldsymbol{\gamma}_v$ is also defined as $\boldsymbol{\gamma}_v = -\mathbf{M}^{-1}\mathbf{G}\hat{\mathbf{p}}$. Similar to (3.1) we obtain

$$\mathbf{G}^T \mathbf{M}^{-1} \mathbf{G} \hat{\mathbf{p}} = \mathbf{G}^T \mathbf{v}_\star^{n+1/2}, \tag{3.2}$$

which we can solve for $\hat{\mathbf{p}}$ and subsequently correct the velocity via $\tilde{\mathbf{v}}^n = \mathbf{v}_\star^n - \mathbf{M}^{-1}\mathbf{G}\hat{\mathbf{p}}$. The difference between (3.1) and (3.2) is that (3.2) computes a divergence-free velocity whereas (3.1) adds an extra term to obtain a non-zero divergence (similar to [53]) in order to correct any drift in volume.

Although the velocity projection is always stable, small time steps and significant volume errors can lead to difficulties as all the missing volume is recovered at once. We alleviate this by introducing a minimum volume recovery time scale $\Delta\tau$ and clamping the last term in (3.1) such that its magnitude is no larger than $\mathbf{V}(\mathbf{x}^0)/\Delta\tau$ in any given time step.

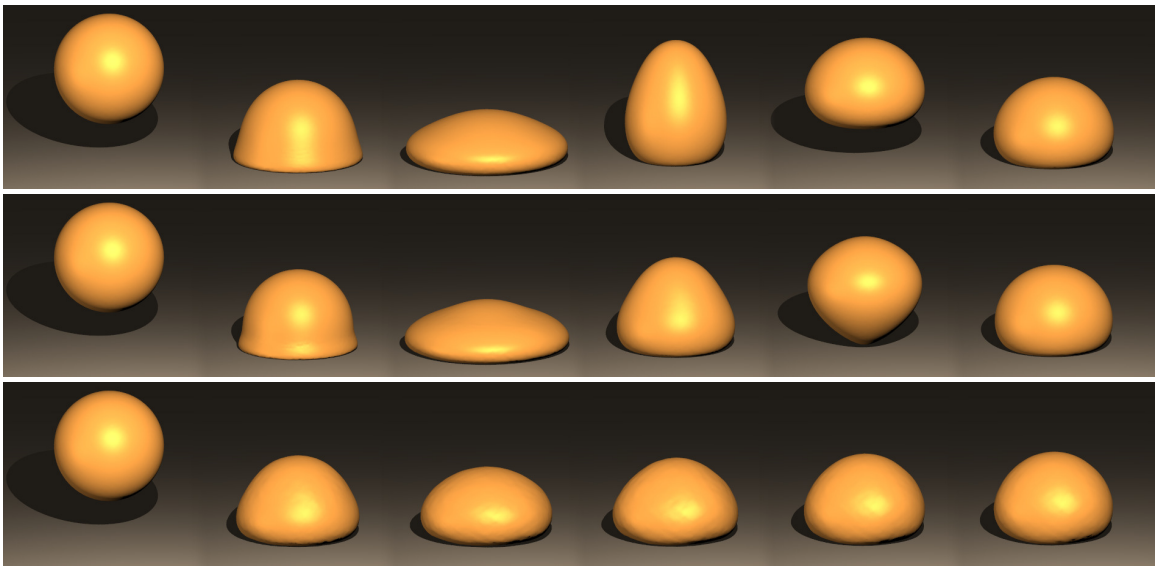


Figure 3.2: An elastic sphere is dropped on the ground. (Top) Enforcing the volume of each one-ring using our method maintains correct volume within 1%. (Middle) Using standard finite element forces with a Poisson's ratio of .45 results in a maximum volume loss of over 15%. (Bottom) Increasing the Poisson's ratio to .499 reduces the maximum volume loss to 2% but causes severe locking of the sphere's degrees of freedom hindering deformation.

3.3 Spatial Discretization

A mesh with n nodes has $3n$ degrees of freedom, and enforcing a volume constraint for each tetrahedron typically results in more than $4n$ constraints (the number of tetrahedra) making the system heavily overconstrained and resulting in locking as shown in Figure 3.2 (bottom). We avoid locking by enforcing incompressibility on one-rings, i.e. on composite elements centered at each node, as shown in the Figure 3.3 (the blue region). This approach adds only n constraints. Composite elements have proven useful in a number of scenarios, see e.g. [154, 23, 119, 41]. Note that the spatial discretization derived below is identical to the average pressure element in [21].

We use a standard finite volume discretization with all information colocated on the nodes of the mesh as in [149]. Let p_0 to p_3 and $\mathbf{x}_0$ to $\mathbf{x}_3$ be the pressures and positions

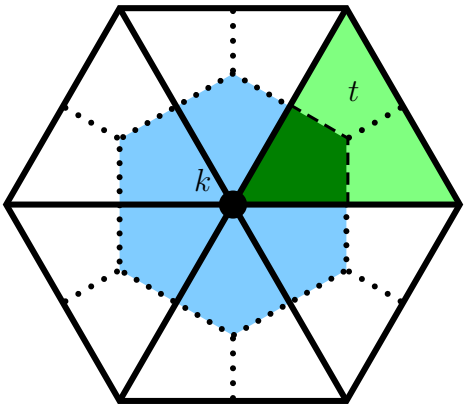


Figure 3.3: The neighborhood of a particle k is the ring of tetrahedra around it. The one-ring is shown in blue with tetrahedron t labeled.

of the four vertices of a tetrahedron. Define

$$\mathbf{D} = \begin{pmatrix} \mathbf{x}_1 - \mathbf{x}_0 & \mathbf{x}_2 - \mathbf{x}_0 & \mathbf{x}_3 - \mathbf{x}_0 \end{pmatrix} \qquad \dot{\mathbf{D}} = \begin{pmatrix} \mathbf{v}_1 - \mathbf{v}_0 & \mathbf{v}_2 - \mathbf{v}_0 & \mathbf{v}_3 - \mathbf{v}_0 \end{pmatrix},$$

and let V be the volume of the tetrahedron, $\mathbf{an}_k$ the outward-facing area-weighted normal opposite vertex k, and

$$\mathbf{B} = V\mathbf{D}^{-T} = -\frac{1}{3}\begin{pmatrix} \mathbf{an}_1 & \mathbf{an}_2 & \mathbf{an}_3 \end{pmatrix}.$$

The linearly interpolated velocity field is

$$\mathbf{v}(\mathbf{x}) = \dot{\mathbf{D}}\mathbf{D}^{-1}(\mathbf{x} - \mathbf{x}_0) + \mathbf{v}_0,$$

from which

$$\nabla \cdot \mathbf{v}(\mathbf{x}) = \mathbf{tr}(\dot{\mathbf{D}}\mathbf{D}^{-1}) = \mathbf{D}^{-T} : \dot{\mathbf{D}}.$$

The total volume-weighted divergence over the one-ring centered at node k is

$$(-\mathbf{G}^T\mathbf{v})_k = \frac{1}{4}\int_{R(k)} \nabla \cdot \mathbf{v}\, d\mathbf{x} = \frac{1}{4}\sum_{t \in R(k)} V_t \mathbf{D}_t^{-T} : \dot{\mathbf{D}}_t = \frac{1}{4}\sum_{t \in R(k)} \mathbf{B}_t : \dot{\mathbf{D}}_t,$$

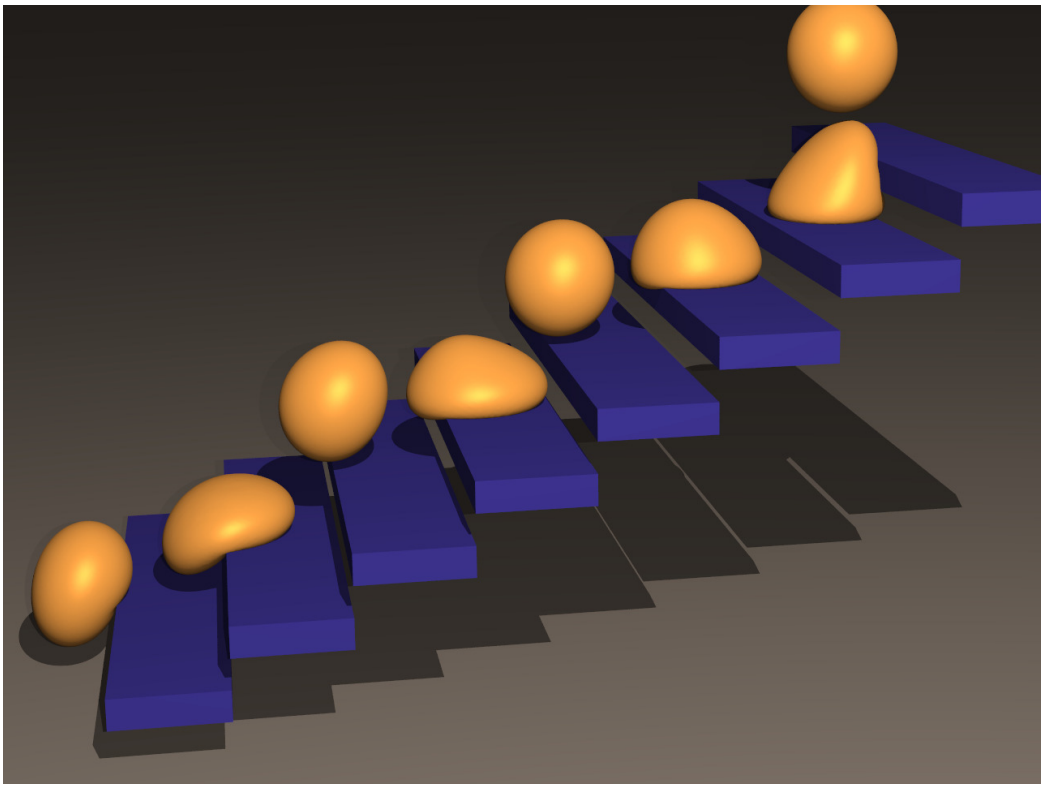


Figure 3.4: An incompressible elastic sphere falls down a flight of stairs illustrating rigid body collisions and contact.

where $R(k)$ is the set of tetrahedra incident on k, and the fact that the divergence of the velocity is constant on each tetrahedron allows us to assign one-fourth of the tetrahedral volume to each incident node.

In line with our use of $-\mathbf{G}^T$ to represent the divergence operator, we construct our gradient operator to be the negative transpose of the divergence operator so that (3.1) and (3.2) result in symmetric positive definite systems allowing for fast iterative techniques such as conjugate gradient. Thus, we want $\langle \nabla p, \mathbf{v} \rangle = \langle p, -\nabla \cdot \mathbf{v} \rangle$. That is,

$$\int_\Omega \nabla p \cdot \mathbf{v}\, d\mathbf{x} + \int_\Omega p \nabla \cdot \mathbf{v}\, d\mathbf{x} = \int_\Omega \nabla \cdot (p\mathbf{v})\, d\mathbf{x} = \int_{\partial\Omega} p\mathbf{v} \cdot d\mathbf{S} = 0$$

This holds, for example, in the case of Dirichlet boundary conditions ($p = 0$ on the boundary). Note that we cannot define gradient using an analogue of the formula

Figure 3.5: An incompressible elastic armadillo drops onto the ground illustrating self-collisions and contact.

used to define divergence, since the resulting forces would not conserve momentum near the boundary.

In order to define $\mathbf{G}$, we assume that the pressure field is zero outside the object (noting that it is straightforward to relax this restriction). We then partition the pressure field into $p = \sum_{t \in R(k)} p_t$, where p_t is a pressure field that agrees with p in t and is identically zero elsewhere. This allows us to restrict attention to a single tetrahedron, since the linearity of the gradient operator gives $(\mathbf{G}p)_k = \sum_{t \in R(k)} (\mathbf{G}p_t)_k$. Each tetrahedron in $R(k)$ makes a contribution to $(-\mathbf{G}^T\mathbf{v})_k$ of the form

$$(-\mathbf{G}^T\mathbf{v})_k = \frac{1}{4}\mathbf{B} : \dot{\mathbf{D}} = -\frac{1}{12}\sum_{j=1}^{3}(\mathbf{v}_j - \mathbf{v}_0) \cdot \mathbf{a}\mathbf{n}_j = -\frac{1}{12}\sum_{j=0}^{3}\mathbf{v}_j \cdot \mathbf{a}\mathbf{n}_j.$$

Thus, we can interpret the divergence operator on a single tetrahedron as a 4×12 matrix

$$-\mathbf{G}^T = -\frac{1}{12}\begin{pmatrix}\mathbf{N}^T \\ \mathbf{N}^T \\ \mathbf{N}^T \\ \mathbf{N}^T\end{pmatrix} \qquad \mathbf{N} = \begin{pmatrix}\mathbf{a}\mathbf{n}_0 \\ \mathbf{a}\mathbf{n}_1 \\ \mathbf{a}\mathbf{n}_2 \\ \mathbf{a}\mathbf{n}_3\end{pmatrix}.$$

We can now write gradient as a 12×4 matrix $\mathbf{G} = \frac{1}{12}\begin{pmatrix}\mathbf{N} & \mathbf{N} & \mathbf{N} & \mathbf{N}\end{pmatrix}$ so that the

contribution of a single tetrahedron to the gradient at a node k is

$$(\mathbf{G}p)_k = \frac{1}{12}\mathbf{N}_k \sum_{j=0}^{3} p_j = \frac{1}{3}\mathbf{an}_k \overline{p},$$

where $\overline{p}$ is the average pressure of the four vertices of the tetrahedron. Summing over $R(k)$ gives

$$(\mathbf{G}p)_k = \frac{1}{3} \sum_{t \in R(k)} \overline{p}_t \mathbf{an}_{t,k},$$

where $\overline{p}_t$ is the average pressure in t and $\mathbf{an}_{t,k}$ is the area-weighted normal of the face opposite node k. This equation is exactly the standard FVM force for a Cauchy stress of $\overline{p}_t$ (see [149]) and can be computed by forming $\mathbf{S} = -\mathbf{B}_t \overline{p}_t$ and distributing the columns of $\mathbf{S}$ to the nodes (where one node gets the negation of the sum of the columns). In particular our volume preservation forces conserve momentum for each tetrahedron independent of other tetrahedra, since the net force on a tetrahedron is

$$\mathbf{F} = \sum_k \mathbf{F}_k = -\sum_k (\mathbf{G}p)_k = -\frac{1}{3}\overline{p} \sum_k \mathbf{an}_k = \mathbf{0}.$$

Let $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}_0$. Then, $\mathbf{an}_1 = -\frac{1}{2}\mathbf{e}_2 \times \mathbf{e}_3$, with similar expressions for $\mathbf{an}_2$ and $\mathbf{an}_3$ obtained by cycling indices. Angular momentum is also conserved per tetrahedron, since the torque is

$$\begin{aligned}
\boldsymbol{\tau} &= \sum_k (\mathbf{x}_k - \mathbf{c}) \times \mathbf{F}_k = \sum_k \mathbf{e}_k \times \mathbf{F}_k + (\mathbf{x}_0 - \mathbf{c}) \times \sum_k \mathbf{F}_k \\
&= -\sum_k \mathbf{e}_k \times (\mathbf{G}p)_k = -\frac{1}{3}\overline{p} \sum_k \mathbf{e}_k \times \mathbf{an}_k \\
&= \frac{1}{6}\overline{p}(\mathbf{e}_1 \times (\mathbf{e}_2 \times \mathbf{e}_3) + \mathbf{e}_2 \times (\mathbf{e}_3 \times \mathbf{e}_1) + \mathbf{e}_3 \times (\mathbf{e}_1 \times \mathbf{e}_2)) = \mathbf{0}
\end{aligned}$$

by Jacobi's identity.

3.4 Collisions and Contact

While steps 2 and 6 of the time integration algorithm work to preserve incompressibility, steps 5, 3, and 9 add additional constraints for object collisions and self-collisions. In practice, we have noticed that the blind application of our algorithm can cause serious artifacts due to these competing constraints, resulting in unusably tangled surfaces. Therefore, we incorporate both object contact and self-contact constraints into our incompressible Poisson equations. Despite being important for robust behavior in the presence of collisions, this coupling is largely undiscussed by previous authors who focused primarily on integrating incompressibility into their particular time or space discretization schemes.

Step 5 sets the position and velocity of particles to respect collisions with objects, and the conjugate gradient solver used in step 7 incorporates constraints in the normal direction to maintain the correct normal velocity for colliding particles, i.e. $\mathbf{n}^T\Delta\mathbf{v} = 0$ where $\mathbf{n}$ is the local unit normal to the collision body and $\Delta\mathbf{v}$ is the change in velocity due to conjugate gradient. We incorporate a similar constraint into the Poisson equations solved in steps 2 and 6 of the algorithm, stressing that this is a linear constraint of the form $\mathbf{c}^T\Delta\mathbf{v} = 0$. Self-contact can similarly be written as linear constraints of the form $\mathbf{c}^T\Delta\mathbf{v} = 0$. Constraining the relative velocity of a point and triangle to not change yields

$$\mathbf{n}^T(\Delta\mathbf{v}_p - w_1\Delta\mathbf{v}_1 - w_2\Delta\mathbf{v}_2 - w_3\Delta\mathbf{v}_3) = 0,$$

where w_i are the barycentric weights of the point on the triangle interacting with particle p and $\mathbf{n}$ is the triangle's normal. Constraining the relative velocity of interacting points in an edge-edge pair yields

$$\mathbf{s}^T((1-\alpha_1)\Delta\mathbf{v}_1 + \alpha_1\Delta\mathbf{v}_2 - (1-\alpha_2)\Delta\mathbf{v}_3 - \alpha_2\Delta\mathbf{v}_4) = 0,$$

where α_i are positions of the interacting points along the segments and $\mathbf{s}$ is the shortest vector between the interacting segments. Self-contact constraints are generated for

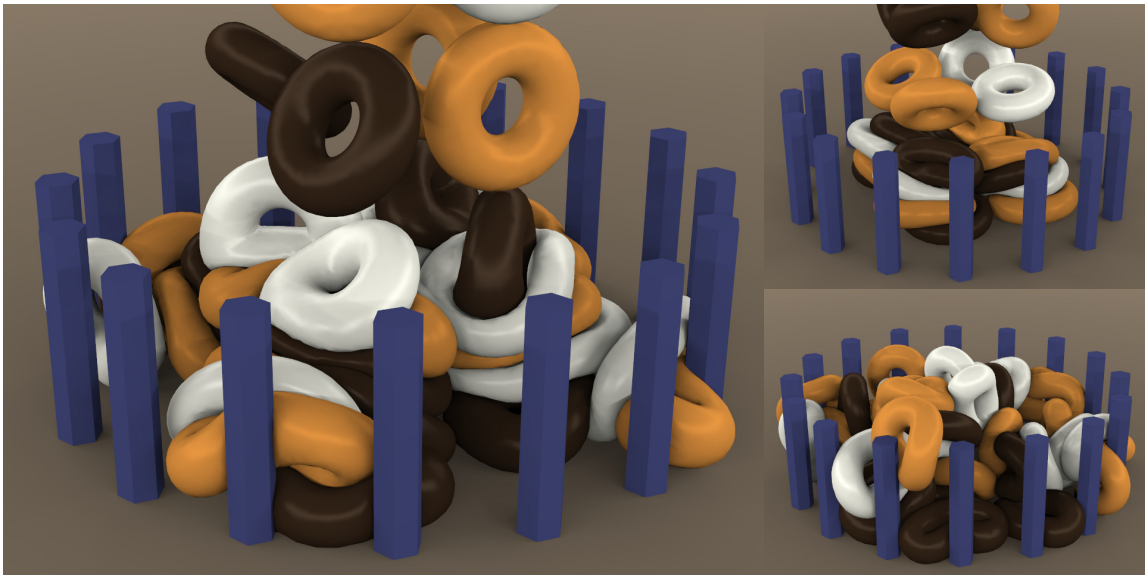


Figure 3.6: 40 incompressible elastic tori fall into a pile illustrating complex collision and contact. Object contact and self-contact are represented as linear constraints during each Poisson solve. Each torus maintains correct volume to within 0.5%, even at the bottom of the pile.

each point-triangle and edge-edge pair currently in close proximity (these correspond to the repulsion pairs in [27]). Note that the ability to set the velocity before the Poisson solve and guarantee no changes during the solve is equivalent to using a Neumann boundary condition on the pressure.

First consider a single constraint, i.e. a single point-object, point-triangle, or edge-edge interaction. We project out any constraint violating contribution by redefining $\boldsymbol{\gamma} = -\mathbf{P}\mathbf{M}^{-1}\mathbf{G}\hat{\mathbf{p}}$, altering the left-hand sides of (3.1) and (3.2) to $\mathbf{G}^T\mathbf{P}\mathbf{M}^{-1}\mathbf{G}\hat{\mathbf{p}}$, where $\mathbf{P}$ projects a change in velocity using an impulse $\mathbf{j}$ defined by

$$\mathbf{P}\Delta\mathbf{v} = \Delta\mathbf{v} + \mathbf{M}^{-1}\mathbf{j}.$$

The impulse $\mathbf{j}$ can be found by minimizing the kinetic energy $\mathbf{j}^T\mathbf{M}^{-1}\mathbf{j}/2$ subject to $\mathbf{c}^T\mathbf{P}\Delta\mathbf{v} = 0$ using the objective function

$$\mathbf{j}^T\mathbf{M}^{-1}\mathbf{j}/2 + \lambda(\mathbf{c}^T\Delta\mathbf{v} + \mathbf{c}^T\mathbf{M}^{-1}\mathbf{j}).$$

Differentiating with respect to $\mathbf{j}$ and setting to zero gives $\mathbf{j} = -\mathbf{c}\lambda$, and substituting this into the constraint equation yields

$$\lambda = (\mathbf{c}^T\mathbf{M}^{-1}\mathbf{c})^{-1}\mathbf{c}^T\Delta\mathbf{v}.$$

Thus,

$$\mathbf{P}\Delta\mathbf{v} = (\mathbf{I} - \mathbf{M}^{-1}\mathbf{c}(\mathbf{c}^T\mathbf{M}^{-1}\mathbf{c})^{-1}\mathbf{c}^T)\Delta\mathbf{v}$$

and

$$\mathbf{P} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{c}(\mathbf{c}^T\mathbf{M}^{-1}\mathbf{c})^{-1}\mathbf{c}^T.$$

Note that $\mathbf{P}\mathbf{M}^{-1}$ is symmetric positive semidefinite with exactly one zero eigenvalue.

In the case of many constraints

$$\mathbf{C}^T\Delta\mathbf{v} = \Big(\mathbf{c}_1 \cdots \mathbf{c}_n\Big)^T\Delta\mathbf{v} = 0,$$

applying the projections in simple Gauss-Seidel order gives $\mathbf{P}_n \cdots \mathbf{P}_1\mathbf{M}^{-1}$ which is only symmetric if none of the constraints overlap. For example, this is violated whenever point-triangle or edge-edge pairs share vertices since the corresponding $\mathbf{P}_i's$ do not commute. One might attempt to alleviate this problem by avoiding sequential application and applying all constraints at once, but this would require inversion of the $n \times n$ matrix $\mathbf{C}^T\mathbf{M}^{-1}\mathbf{C}$ appearing in the general form of the projection

$$\mathbf{P} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{C}(\mathbf{C}^T\mathbf{M}^{-1}\mathbf{C})^{-1}\mathbf{C}^T,$$

which is prohibitively expensive for complex scenarios with dynamic constraints. Instead, we apply the projections in alternating forward and backward Gauss-Seidel sweeps using the symmetric positive semidefinite matrix

$$\mathbf{P}\mathbf{M}^{-1} = (\mathbf{P}_1 \cdots \mathbf{P}_n \cdots \mathbf{P}_1)^q\mathbf{M}^{-1},$$

where q is a small integer. To see that this is symmetric, consider the case with $n = 2$ and $q = 1$. Since $\mathbf{M}^{-1}$ and $\mathbf{P}_k\mathbf{M}^{-1}$ are symmetric, $\mathbf{P}_k\mathbf{M}^{-1} = (\mathbf{P}_k\mathbf{M}^{-1})^T$ or

$\mathbf{P}_k\mathbf{M}^{-1} = \mathbf{M}^{-1}\mathbf{P}_k^T$. Then,

$$\mathbf{P}\mathbf{M}^{-1} = (\mathbf{P}_1\mathbf{P}_2\mathbf{P}_1)\mathbf{M}^{-1} = \mathbf{P}_1\mathbf{P}_2\mathbf{P}_2\mathbf{P}_1\mathbf{M}^{-1} = \mathbf{P}_1\mathbf{P}_2\mathbf{P}_2\mathbf{M}^{-1}\mathbf{P}_1^T = (\mathbf{P}_1\mathbf{P}_2)\mathbf{M}^{-1}(\mathbf{P}_1\mathbf{P}_2)^T$$

Similarly, if $n = q = 2$, then

$$\mathbf{P}\mathbf{M}^{-1} = (\mathbf{P}_1\mathbf{P}_2\mathbf{P}_1)^2\mathbf{M}^{-1} = \mathbf{P}_1\mathbf{P}_2\mathbf{P}_1\mathbf{M}^{-1}\mathbf{P}_1^T\mathbf{P}_2^T\mathbf{P}_1^T = (\mathbf{P}_1\mathbf{P}_2\mathbf{P}_1)\mathbf{M}^{-1}(\mathbf{P}_1\mathbf{P}_2\mathbf{P}_1)^T.$$

In particular, note that $\mathbf{P}\mathbf{M}^{-1} = \mathbf{P}\mathbf{M}^{-1}\mathbf{P}^T$, so that the system could alternatively be formulated as $\mathbf{G}^T\mathbf{P}\mathbf{M}^{-1}\mathbf{P}^T\mathbf{G}\hat{\mathbf{p}} = (\mathbf{P}^T\mathbf{G})^T\mathbf{M}^{-1}(\mathbf{P}^T\mathbf{G})\hat{\mathbf{p}}$, so that the velocity projection and forward-backward sweeps could instead be replaced with a modified gradient operator, and the projections imposed could be interpreted as a boundary condition.

Applying a single unsatisfied projection $\mathbf{P}_i$ strictly reduces the energy, so this iteration is stable and always converges to the correct constraint satisfying velocity. In practice, we found that using only $q = 4$ iterations reduced the constraint violating components by 1-2 orders of magnitude on average. Since the pressure system is itself solved to only 1% accuracy and the alternating sweeps ensure symmetry of the full matrix regardless of convergence, this was sufficient for all our examples. These projections increase the cost of solving for pressure only moderately since there are typically many fewer collisions than vertices and each $\mathbf{P}_i$ can be applied in constant time. Since object and point-triangle contacts have more coherent normals and are typically better behaved than edge-edge contacts, we bias unconverged results towards the former by placing them first (and last) in the ordering.

The final matrix $\mathbf{G}^T\mathbf{P}\mathbf{M}^{-1}\mathbf{G}\hat{\mathbf{p}}$ is always symmetric positive semidefinite, but it can become singular in cases with large numbers of constraints. Since conjugate gradient breaks down for singular matrices, we instead use MINRES, an alternative Krylov space method which requires only symmetry of the matrix. MINRES required significantly fewer iterations than conjugate gradient even in nonsingular cases, though this is largely due to MINRES targeting the minimization of the terminating criterion. For example, Figure 3.2 averaged 72 iterations with conjugate gradient and 34 iterations with MINRES (with small additional cost per iteration). Thorough description and

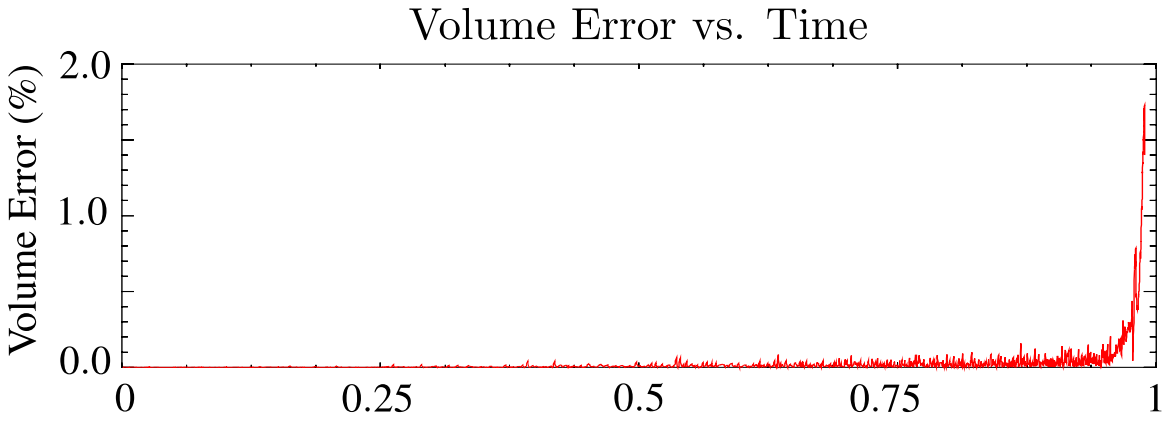


Figure 3.7: The volume error, in percent, as a sphere is pressed between plates is shown. The plates are just touching the sphere at time 0 and move towards each other with constant velocity until they meet at time 1. The sphere is flattened into a pancake with a width only 1.4% of its original diameter before a volume error of one percent is obtained.

analysis of MINRES, conjugate gradient, and related solvers for singular or nearly singular systems can be found in [35].

3.5 Examples

We used the method of [81] for internal deviatoric finite element forces in all our examples. When necessary, we used a minimum volume recovery time scale of one-fifth of a frame. Figure 3.2 shows a comparison of our method against a standard finite volume discretization using a 104k element mesh. Using a 3GHz Xeon machine, the computational cost was 18 s/frame for our method, 25 s/frame with Poisson's ratio .45, and 3.4 min/frame with Poisson's ratio .499. Similarly the simulation time for Figure 3.4 was 34 s/frame. The armadillo simulations in Figures 3.1 and 3.5 were both under 4 min/frame with a 112k element mesh. The simulation in Figure 3.6 took an average of 15 min/frame for 40 12k element meshes (500k elements total) with approximately 65% the time spent in the two Poisson solves due to the complexity of the contact constraints.

As a stress test for our method, we squeezed an incompressible sphere with 22k elements between two kinematic plates (similar to an example in [74]). The minimum volume recovery time scale was not used (i.e. we set $\tau = 0$). The sphere was successfully compressed to 1.1% of its original thickness before numerical error forced the time step to zero (all computations were performed in single precision). The total volume error remained below 1.7% throughout the simulation, and was lower than 0.1%, 0.5%, and 1% until the sphere reached 13%, 2.3%, and 1.4% of its original thickness, respectively. A plot of volume error vs. time is shown in Figure 3.7. For this simulation we modified the time integration scheme to enforce contact constraints during both backward Euler solves (steps 1 and 7) instead of only in step 7, so that the volume correction in step 2 used the correct collision-aware velocities for particles in contact with the plates. The use of uncorrected velocities as input to volume correction would have caused significant degradation for this simulation due to the high tension involved. This modification was not necessary for the other examples, which is fortunate since enforcing contact constraints in both solves typically causes sticking artifacts during separation.

3.6 Alternate Formulation

The method presented for applying incompressibility as a correction step to an implicit velocity update. This problem can also be viewed as updating velocities subject to the two constraints, the incompressibility constraint and the contact constraint. In the position update, the incompressibility constraint takes the form

$$\mathbf{G}^T\mathbf{v} = (\mathbf{V}(\mathbf{x}^n) - \mathbf{V}(\mathbf{x}^0))/\Delta t, \tag{3.3}$$

so that a nonzero divergence is targeted. When cast in this way the SPD framework to be developed in Chapter 5 can be used to merge both steps into a single SPD system. That is, steps 1 and 2 can be combined into a single coupled solve, and steps 6 and 7 can be combined into a coupled solve. In such a formulation, forward

and backward projection sweeps become unnecessary. This formulation is developed in Section 5.10.

3.7 Conclusion

We proposed a novel technique for enforcing local incompressibility in deformable solids by drawing ideas from computational fluid dynamics. We benefit from the simplicity and flexibility of tetrahedra while avoiding the pitfalls of locking by enforcing volume preservation over one-rings instead of individual tetrahedra. We augmented our method to incorporate both object contact and self-contact constraints into the incompressible solve to alleviate problems with conflicting constraints.

Chapter 4

Rigid Deformable Coupling

The previous two chapters considered rigid bodies and deformable bodies where only one is evolved in time. In this chapter, we develop a method for fully two-way monolithic coupling of rigid and deformable bodies. We combine the time integration schemes for each into a single integration scheme with fully coupled linear systems. The scheme includes fully coupled collisions and contact as well as support for the incompressible solids from Chapter 3, articulation, and controllers. We also introduce constraints in the linear systems for contact and articulation to strengthen the coupling. This section is based on work published in [134].

4.1 Introduction

A number of authors have proposed methods for simulating two-way coupling between rigid and deformable bodies, see e.g. [11, 115, 84, 97, 138]. Although coupling deformable and rigid bodies is interesting from the standpoint of simulating new phenomena as demonstrated by those authors, it is also quite interesting from the standpoint of designing creatures. Typically creatures are designed as articulated rigid bodies with some sort of joint or muscle control with notable examples being

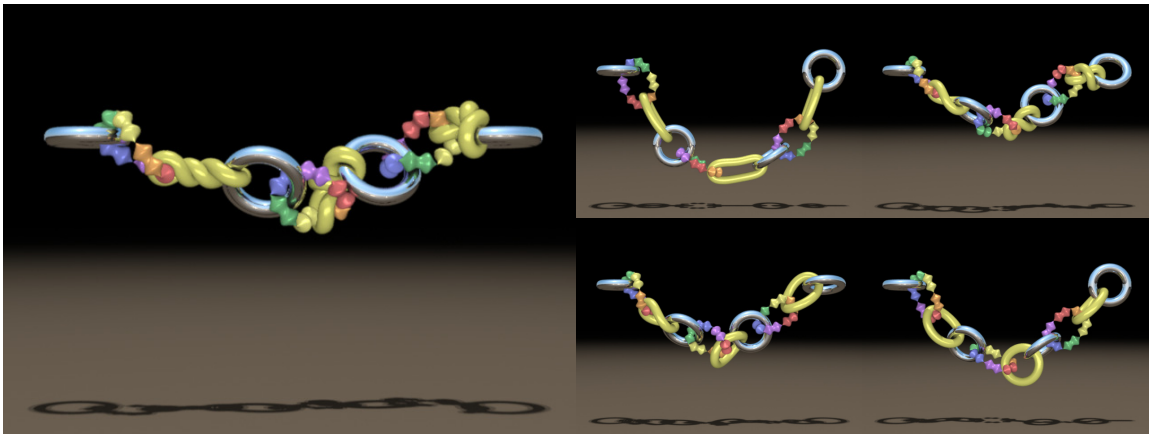


Figure 4.1: A chain is assembled from rigid and deformable tori as well as loops of articulated rigid bodies. The leftmost torus is static (kinematic and fixed), and the rightmost torus is kinematically spun to wind and unwind the chain.

Luxo Jr. [171], athletes from the 1996 Summer Olympics [73], and the virtual stunt man of [50]. However, creatures are more life-like when their internal skeleton can be used to drive a deformable exterior, such as the tentacles for Davy Jones [40, 167]. The difficulty with wrapping a rigid skeleton with a deformable exterior is that the creature can only interact with its environment if environmental forces deform the exterior and the deformable exterior subsequently applies forces to the rigid interior. That is, modeling deformable creatures with rigid skeletons requires two-way rigid/deformable interaction [56].

For the two-way coupling of physical phenomena, there are two common approaches. One approach is to start with the best methods for each phenomenon and subsequently design methods for linking these disparate simulation techniques together. Typically interleaving is necessary, where each simulation is run using the results of the previous one in an alternating one-way coupled fashion. This leads to stability issues as one system cannot adequately predict what the other system will do–similar to explicit time integration. Various ad hoc tricks can be applied to increase the stability making the system semi-implicit (e.g. solving for the effect of damping forces between rigid and deformable bodies implicitly and subsequently mapping the momentum back to the rigid body simulator [138]). Still, stability issues remain. The

other approach is to design a fully two-way coupled system, and there are essentially two ways to accomplish this. One could use the same type of simulation for both types of phenomena (e.g. SPH for both solids and liquids [105] or cloth as an articulated net of rigid bodies [17]). Unfortunately, this typically leads to inferior methods for at least one of the two phenomena being simulated. For example, one could imagine extending the mass-spring simulation framework to the rigid limit and simulating rigid bodies with very stiff springs. We take the alternative approach and fully hybridize the simulation frameworks in a manner that maintains the ability to use the more advanced techniques for each physical phenomenon. That is, we want to retain the ability of our rigid body framework to accurately model contact, collision, stacking, friction, articulation, and proportional derivative (PD) control and our deformable object framework to handle arbitrary constitutive models, finite elements, masses and springs, volumes or thin shells, contact, collision, and self-collisions. In addition, we would like to maintain stability and avoid ad hoc methods for interleaving the simulations.

Full two-way coupling of state-of-the-art rigid body and deformable object time integration schemes, though seemingly straightforward, is more difficult and subtle than it appears. Robust rigid body systems tend to be sensitive to the time integration scheme because a minor change in the scheme can, for example, cause blocks to tumble rather than slide down an inclined plane. In contrast, deformable object Newmark schemes are far less sensitive and allow more freedom in the way collisions are processed but utilize separate position and velocity updates as well as implicit solves. Our approach demonstrates that we can fully integrate these disparate methods without losing their individual capabilities. Notably, this can be done by coupling together standard rigid body and standard deformable body simulation systems using our newly proposed techniques instead of requiring one to design a unique simulation system from scratch.

We propose a unified time integration scheme, where the rigid and deformable bodies are integrated forward together in every manner, not only with position and velocity evolution, but such that collision, contact, and interpenetration resolution are all

handled at the fine-grained level with fully two-way coupled algorithms. On the rigid body side, we were guided by the time integration scheme proposed by [64], which proposes a clean separation of contact and collision and handles both simple and more complex scenarios ranging from a block sliding down an inclined plane to large-scale stacking behaviors. We also wanted our rigid body simulator to include the effects of articulation and internally torque-controlled joints [169, 168]. On the deformable side, we require a Newmark-style algorithm as in [138], which cleanly separates the evolution of position and velocity. The deformable algorithm should at least be implicit in the damping forces as in [27] but could also be fully implicit as in [12]. We propose such a time integration scheme, which evolves every object synchronously using fully coupled algorithms at each step.

We describe in detail the steps of our unified time integration scheme below. We note, however, that the approach does not depend on our particular choice of algorithms for the various subsystems but rather serves as a proof of concept of the benefits of a unified approach. Other choices of specific algorithms such as contact and collision can be substituted. Much of what we incorporate is optional, including the incompressibility of [80], the articulation of [169], the PD control of [168], and the bindings of [138]. We include them to demonstrate the breadth of phenomena that can be incorporated, but we note that the remaining system remains fully functional in their absense.

4.2 Time Integration

The basic structure of our time integration scheme is that of a Newmark method. Newmark methods are characterized by separate position and velocity updates. In particular, the velocity used in the position update can be distinct from the final velocity, and we take advantage of this to treat the position and velocity updates differently. For example, one might add constraint violating components to the velocity during the position update to correct drift while projecting out these components

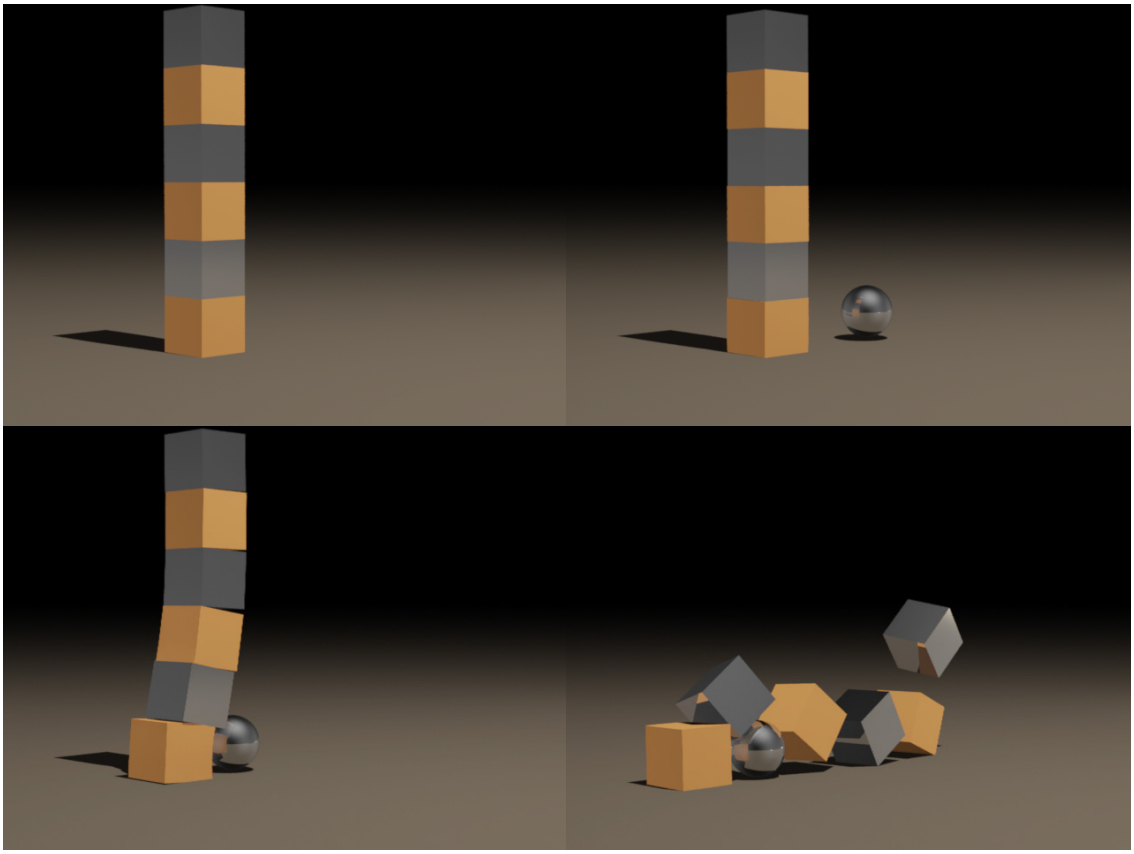


Figure 4.2: Silver rigid boxes and orange deformable boxes are arranged to form a stack, which is knocked down by a rigid ball.

in the final velocity update (e.g. as in [80]). Moreover, maintaining the Newmark structure allows us to incorporate a wide range of algorithms from computational mechanics. For example, we can enforce the incompressibility of materials, deal with contact and collisions, self-collisions, friction, etc. For background on the Newmark family of methods see e.g. [76].

We begin by outlining our time integration scheme as composed of five major steps:

I. Advance velocity $\mathbf{v}^n \to \tilde{\mathbf{v}}^{n+\frac{1}{2}}$

II. Apply collisions $\mathbf{v}^n \to \hat{\mathbf{v}}^n$, $\tilde{\mathbf{v}}^{n+\frac{1}{2}} \to \hat{\mathbf{v}}^{n+\frac{1}{2}}$

III. Apply contact and constraint forces $\hat{\mathbf{v}}^{n+\frac{1}{2}} \to \mathbf{v}^{n+\frac{1}{2}}$

IV. Advance positions $\mathbf{x}^n \to \mathbf{x}^{n+1}$ using $\mathbf{v}^{n+\frac{1}{2}}$, $\hat{\mathbf{v}}^n \to \overline{\mathbf{v}}^n$

V. Advance velocity $\overline{\mathbf{v}}^n \to \mathbf{v}^{n+1}$

Adhering to the time integration scheme proposed in [64], we cleanly separate position and velocity updates from contact and collision. Collisions are done with time t^n velocities so that objects in contact do not bounce, and contacts are performed only on a subset of pairs processed by collisions so that one does not inaccurately apply contact forces to objects before they collide. The algorithmic ordering of collisions followed by contact followed by position updates followed by velocity updates is exactly as in [64], except for the initial advancement of velocities to $\mathbf{v}^{n+1/2}$. In fact, their algorithm would have been more accurate if it had done this rather than using the full Δt for the predictive velocities, obtaining $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^n + \Delta t^2 \mathbf{a}$. [168] took special measures to deal with this inaccuracy, changing parameters to get the desired solution. Using $\mathbf{v}^{n+1/2}$ as in the Newmark scheme automatically alleviates this issue.

Each of the five major steps of the time integration scheme is further described in the five sections that follow.

4.3 Step I: Advance Velocity

Typically the first step in a Newmark iteration scheme is to predict the velocity that will be used to update the positions. We accomplish this as follows:

1. Advance velocities $\mathbf{v}_\star^{n+\frac{1}{2}} = \mathbf{v}^n + \frac{\Delta t}{2}\mathbf{a}(t^{n+\frac{1}{2}}, \mathbf{x}^n, \mathbf{v}_\star^{n+\frac{1}{2}})$
2. Apply volume correction $\mathbf{v}_\star^{n+\frac{1}{2}} \to \mathbf{v}_{\star\star}^{n+\frac{1}{2}}$
3. Apply self-repulsions $\mathbf{v}_{\star\star}^{n+\frac{1}{2}} \to \tilde{\mathbf{v}}^{n+\frac{1}{2}}$

Here and in the remainder of the paper we use starred variables to denote intermediate states that do not carry over from section to section. When using semi-implicit time integration with implicit, linear damping forces as in [28], step 1 requires the

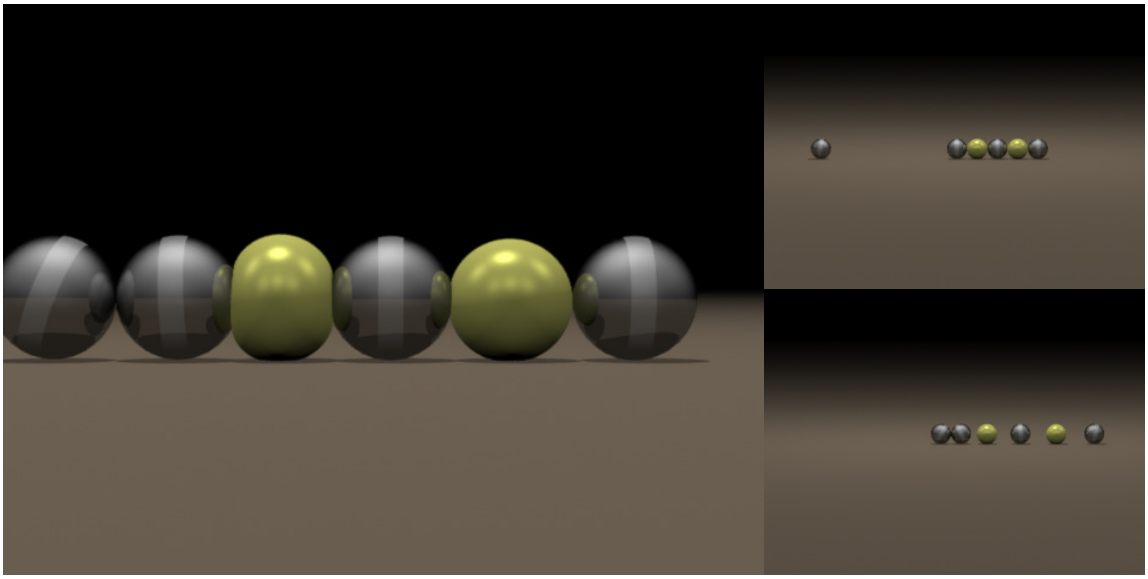


Figure 4.3: A row of silver rigid balls and yellow deformable balls are impacted by a rigid ball. The yellow balls are modeled as deformable and incompressible.

solution of a linear system, which we solve using the conjugate gradient method. One could instead use the fully implicit method from [12]. In that case step 1 becomes $\mathbf{v}_\star^{n+1/2} = \mathbf{v}^n + \frac{\Delta t}{2}\mathbf{a}(t^{n+1/2}, \mathbf{x}^{n+1/2}, \mathbf{v}_\star^{n+1/2})$, where $\mathbf{x}^{n+1/2}$ is replaced with $\mathbf{x}^{n+1/2} = \mathbf{x}^n + \frac{\Delta t}{2}\mathbf{v}_\star^{n+1/2}$. The resulting nonlinear equation is subsequently solved via Newton-Raphson iteration. During the solve, the acceleration in step 1 is projected to satisfy equality constraints such as joint velocity equality constraints, but we avoid applying constraints that may lead to undesirable sticking artifacts at this stage.

If we are simulating incompressible solids, as in Figure 4.3, step 2 adjusts the velocities such that local volume errors will be corrected when positions are advanced [80]. One could also make the velocity field inextensible as in [59]. Step 3 applies self-repulsions as in [27].

While we include steps 2 and 3 here, the only essential part of Step I is to advance the velocity to time $t^{n+1/2}$ as required by the Newmark structure.

Figure 4.4: Ten rigid balls and ten deformable tori fall on a cloth trampoline. The trampoline is constructed by embedding the cloth in a kinematically controlled torus using the framework of [138]. The trampoline tosses the objects after they have landed and then allows them to settle.

4.4 Step II: Collisions

Once the time $t^{n+1/2}$ velocities have been computed, we correct them for rigid/rigid, rigid/deformable and deformable/deformable collisions. As in [64], collisions are processed before contact to allow elastically colliding rigid bodies to bounce. The main goal of this step is to adjust velocities for collisions, and other collision algorithms may be used. The steps we use for our collision processing are

1. Process collisions $\mathbf{v}^n \to \mathbf{v}^n_\star$
2. Apply post-stabilization $\mathbf{v}^n_\star \to \hat{\mathbf{v}}^n$
3. Re-evolve velocities $\hat{\mathbf{v}}^{n+\frac{1}{2}} = \hat{\mathbf{v}}^n + (\tilde{\mathbf{v}}^{n+\frac{1}{2}} - \mathbf{v}^n)$

For collision detection, all positions are evolved forward in time to look for interferences between pairs of rigid/rigid, rigid/deformable, or deformable/deformable bodies using the update formula $\mathbf{x}^{n+1}_\star = \mathbf{x}^n + \Delta t \tilde{\mathbf{v}}^{n+\frac{1}{2}}$ (see Section 4.4.1 for rigid body orientations). As in [64], collisions are processed using $\mathbf{v}^n$ so that objects in contact do not bounce, even if they have a nonzero coefficient of restitution. For details on rigid/rigid collisions, see [64]. Rigid/deformable collisions are processed by iteratively colliding

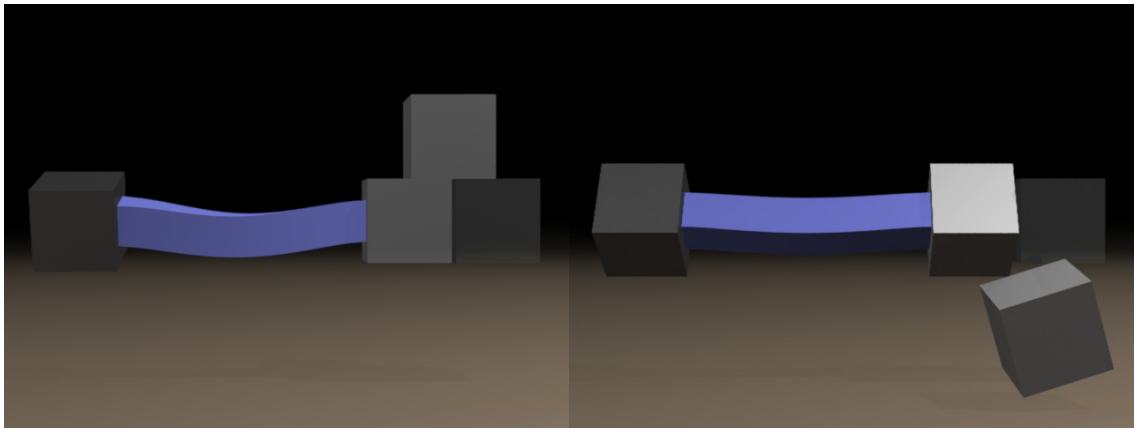


Figure 4.5: The four silver boxes are rigid, and the deformable blue bar is attached to the boxes using the embedding framework of [138]. The far left box is kinematically rotated, causing the bar to twist and subsequently turn the box to its right, which is attached to the static box to the right of it by a twist joint. The free box on top falls off as the bar is twisted.

each particle of the deformable body against the rigid body. Each rigid/particle pair is processed using the rigid/rigid collision algorithm treating the particle as a rigid body (with an infinite inertia tensor so that it does not rotate) and a zero coefficient of restitution. For deformable/deformable pairs, one could use tetrahedral collisions as in [150]; however, we instead use the self-collisions of [27] after the position update as described in Section 4.6.

In Gauss-Seidel fashion, we iteratively process the collisions one pair at a time and re-update the post-collision position of each body via $\mathbf{x}_\star^{n+1} = \mathbf{x}^n + \Delta t(\mathbf{v}_\star^n + (\tilde{\mathbf{v}}^{n+1/2} - \mathbf{v}^n))$, where $\tilde{\mathbf{v}}^{n+1/2} - \mathbf{v}^n$ includes the forces added in step 1 of Section 4.3. If no collisions were applied, then $\mathbf{v}_\star^n = \mathbf{v}^n$ and the position is unchanged. Note that the positions computed here are only used to resolve collisions and are subsequently discarded. We also explicitly save a list of all pairs processed so that contact later considers only those pairs already processed by collisions. This ensures that newly colliding objects will have an opportunity to bounce before being processed for contact. After processing collisions, we apply the post-stabilization algorithm of [169] to ensure that the collision velocities do not violate joint constraints. Finally, step 3 applies the effects of collisions $\hat{\mathbf{v}}^n - \mathbf{v}^n$ to the time $t^{n+1/2}$ velocities $\tilde{\mathbf{v}}^{n+1/2}$. We consider an

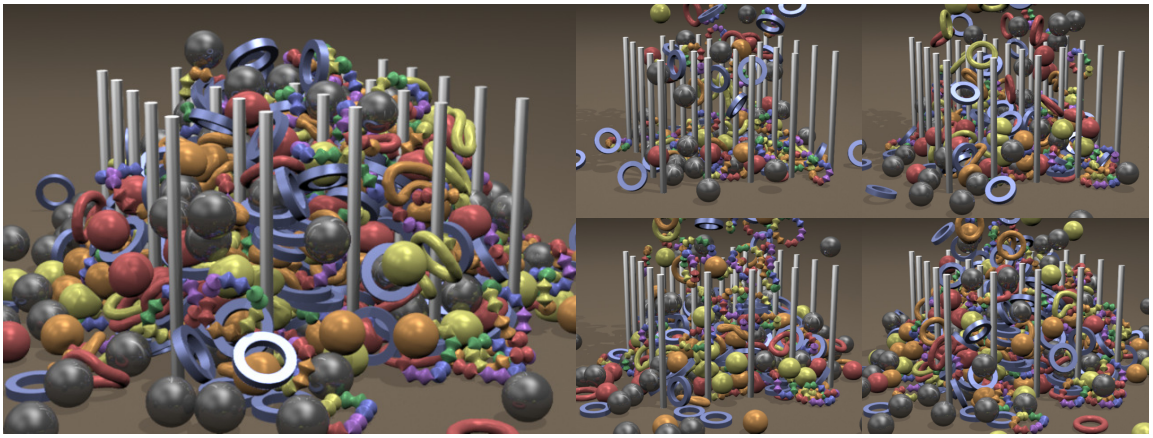


Figure 4.6: 1600 bodies are dropped on a 5×5 grid of static rigid pegs to form a pile. There are 160 colored deformable balls, 160 colored deformable tori, 160 rainbow-colored articulated loops with six rigid bodies each, 160 silver rigid balls and 160 blue rigid rings.

alternative global formulation of post-stabilization in Appendix C.

4.4.1 Second Order Rigid Body Evolution

Given a rigid body in a certain orientation with a certain kinetic energy and angular momentum, the number of states the rigid body is allowed to evolve to in the absence of external torque is limited by conservation of both energy and momentum. Not every orientation will conserve both quantities, and typical simulators only conserve momentum, allowing kinetic energy to rise or fall. [169] used the first order update

$$\mathbf{R}^{n+1} = e^{\Delta t \boldsymbol{\omega}^*} \mathbf{R}^n,$$

where $\boldsymbol{\omega}^*$ is the matrix such that $\boldsymbol{\omega}^* u = \boldsymbol{\omega} \times \mathbf{u}$ for any vector $\mathbf{u}$. The matrix $e^{\Delta t \boldsymbol{\omega}^*}$ is a matrix exponential and corresponds to a rotation of $\Delta t \|\boldsymbol{\omega}\|$ about the axis $\boldsymbol{\omega}$. To reduce errors in kinetic energy, we instead propose using the second order update

$$\mathbf{R}^{n+1} = e^{\Delta t \mathbf{u}^*} \mathbf{R}^n \qquad \mathbf{u} = \Delta t \boldsymbol{\omega} + \frac{1}{2} \Delta t^2 \mathbf{I}^{-1} \mathbf{L}^* \boldsymbol{\omega}.$$

A straightforward Taylor expansion can be used to verify that this formula is second order accurate. Rather than working out the Taylor series, which is rather tedious, we instead note that the result can be obtained by considering an update rule similar to second order Runge Kutta. Higher order corrections are also possible, and the third order variant is

$$\mathbf{u} = \Delta t\boldsymbol{\omega} + \frac{1}{2}\Delta t^2\mathbf{I}^{-1}\mathbf{L}^*\boldsymbol{\omega} + \frac{1}{12}\Delta t^3(\boldsymbol{\omega}^*\mathbf{I}^{-1} - 2\mathbf{I}^{-1}\boldsymbol{\omega}^* + 2\mathbf{I}^{-1}\mathbf{L}^*\mathbf{I}^{-1})\mathbf{L}^*\boldsymbol{\omega}.$$

For a derivation of the second and third variants, see Appendix A.

4.5 Step III: Contact and Constraint Forces

The main purpose of this section is contact and constraint processing. Articulation and PD are optional, and other contact processing algorithms could be substituted. The steps used for the contact and constraint forces stage are

1. Compute contact graph using $\hat{\mathbf{v}}^n$ and $\hat{\mathbf{v}}^{n+\frac{1}{2}}$
2. Apply PD to velocities $\hat{\mathbf{v}}^{n+\frac{1}{2}} \rightarrow \mathbf{v}_\star^{n+\frac{1}{2}}$
3. Apply contact and pre-stabilization $\mathbf{v}_\star^{n+\frac{1}{2}} \rightarrow \mathbf{v}^{n+\frac{1}{2}}$

We compute a contact graph for the rigid bodies similar to [64], which also includes the order in which articulated rigid bodies will be processed for pre-stabilization as in [169]. We also apply PD control as in [168]. Proceeding in the order determined by the contact graph, each rigid body is processed in turn for contact with all rigid and deformable bodies it interpenetrates. For rigid/rigid pairs, we perform contact as in [64]. Rigid/deformable pairs are treated in the same manner as they were treated for collisions, since we always treat deformable body collisions inelastically. This process is iterated as in [64]. If one were using tetrahedral collisions for deformable/deformable pairs [150], they would also be processed at this stage. We instead apply the self-collisions of [27] after the position update as described in Section 4.6 to resolve deformable/deformable contact. The stack in Figure 4.2

demonstrates the effectiveness of the two-way contact algorithm, standing upright until being struck at its base by a rigid ball.

4.5.1 Improved Pre-stabilization

[169] performed pre-stabilization on orientations by solving a quaternion equation of the form $f(\mathbf{j}_\tau) = \mathbf{q}_p(\mathbf{j}_\tau) - \mathbf{q}_c(\mathbf{j}_\tau) = \mathbf{0}$ using Newton-Raphson iteration. Unit quaternions represent orientations with a two-to-one relationship, so that the overall sign of a quaternion is not significant. This calls into question the meaning of performing a quaternion subtraction, and we have indeed observed this to cause problems. To alleviate issues with quaternion subtraction, we instead solve $\mathbf{g}(\mathbf{j}_\tau) = \mathbf{q}_p\mathbf{q}_c^{-1} = (\pm 1, \mathbf{0})$. $\mathbf{g}(\mathbf{j}_\tau)$ consists of both a scalar and vector part, and it turns out to be enough to solve the nonlinear equation that sets the vector part equal to $\mathbf{0}$. This neatly avoids the sign ambiguity entirely. We consider an alternative global formulation in Appendix C.

4.6 Step IV: Advance Positions

In the preceding steps, we made all the desired adjustments to the velocities. The purpose of Step IV is then to evolve the bodies to their final positions, which we do as follows:

1. Advance positions $\hat{\mathbf{x}}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^{n+\frac{1}{2}}$
2. Interpenetration resolution $\hat{\mathbf{x}}^{n+1} \rightarrow \mathbf{x}^{n+1}$
3. Post-stabilization $\hat{\mathbf{v}}^n \rightarrow \overline{\mathbf{v}}^n$

Since our rigid/rigid and rigid/deformable contact algorithms do not guarantee that the bodies are completely interpenetration free, we apply an interpenetration resolution method (similar in spirit to [10]), as described in detail below and in Appendix B. Finally, since steps 1 and 2 changed rigid body orientations, rigid body angular velocities also changed, and thus post-stabilization of velocities for articulated rigid bodies must be applied.

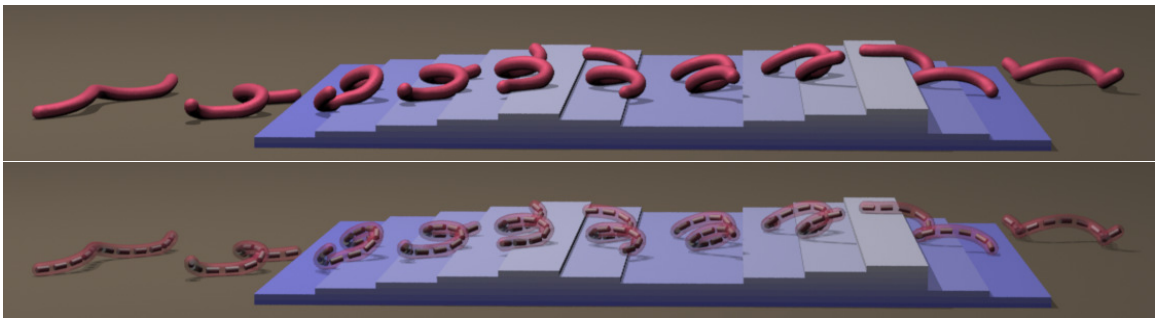


Figure 4.7: A snake is constructed by embedding a 12-joint articulated rigid body skeleton in a deformable body. The snake sidewinds up and down stairs using PD control on its skeleton.

4.6.1 Interpenetration Resolution

One might consider an approach to resolving penetration that operates by iterating pairwise algorithms. This approach suffers from the *messenger problem*, where a large amount of momentum must be exchanged through low-capacity messengers. Consider a row of colliding objects, with the outer two objects being massive and the objects between them being light. The collision must be resolved by moving the massive outer bodies, which requires the exchange of a large impulse. Because the objects in between are light, they are unable to transfer large impulses without obtaining high velocities. Each collision processing step is only able to transfer an impulse across the chain one body at a time, so a large number of iterations are required to complete the momentum transfer.

With this in mind, we propose a novel method to remove small interpenetrations between bodies. Rather than process pairwise as was done for the contact and collision algorithms, we consider one central object (either a rigid body or a particle of a deformable body) at a time and push out every outer object (rigid or deformable) intersecting it simultaneously in a fully two-way coupled fashion. This helps address the messenger problem by allowing a single step to transfer impulses two links at a time instead of one and by avoiding the problem entirely when there is only one such light body. This approach is not easily applied to the full contact and collision problems, primarily because of the complications introduced by friction. The central

body c exchanges an impulse $\mathbf{j}_o$ with an outer body o resulting in a change in relative velocity at the pair's intersection point of

$$m_c^{-1}\mathbf{j} + \mathbf{r}_o^{*T}\mathbf{I}_c^{-1}\mathbf{j}_\tau - \mathbf{K}_o\mathbf{j}_o = \mathbf{v}^{n+1}, \tag{4.1}$$

where

$$\mathbf{K}_o = m_o^{-1}\boldsymbol{\delta} + \mathbf{q}_o^{*T}\mathbf{I}_o^{-1}\mathbf{q}_o^*$$

is the impulse factor, a sort of generalized inverse mass, and $\mathbf{r}_o$ and $\mathbf{q}_o$ are the vectors from the center of masses of body c and body o to the point of application of $\mathbf{j}_o$, respectively, and $\mathbf{j} = -\sum_o \mathbf{j}_o$ and $\mathbf{j}_\tau = -\sum_o \mathbf{r}_o^*\mathbf{j}_o$ are the net linear and angular impulses applied to body c. Solving Equation (4.1) for $\mathbf{j}_o$ and plugging the result into the expressions for $\mathbf{j}$ and $\mathbf{j}_\tau$ gives the symmetric 6×6 system

$$\begin{pmatrix} m_c\boldsymbol{\delta} + \sum\limits_o \mathbf{K}_o^{-1} & \sum\limits_o \mathbf{K}_o^{-1}\mathbf{r}_o^{*T} \\ \sum\limits_o \mathbf{r}_o^*\mathbf{K}_o^{-1} & \mathbf{I}_c + \sum\limits_o \mathbf{r}_o^*\mathbf{K}_o^{-1}\mathbf{r}_o^{*T} \end{pmatrix} \begin{pmatrix} m_c^{-1}\mathbf{j} \\ \mathbf{I}_c^{-1}\mathbf{j}_\tau \end{pmatrix} = \begin{pmatrix} \sum\limits_o \mathbf{K}_o^{-1}\mathbf{v}^{n+1} \\ \sum\limits_o \mathbf{r}_o^*\mathbf{K}_o^{-1}\mathbf{v}^{n+1} \end{pmatrix}. \tag{4.2}$$

Once Equation (4.2) is solved for the net impulse $\mathbf{j}$ and $\mathbf{j}_\tau$ on the central body, each $\mathbf{j}_o$ is obtained from substitution into Equation (4.1). To achieve a desired separation distance $\mathbf{d}$, we take $\mathbf{v}^{n+1} = \mathbf{d}/\Delta\tau$ and integrate positions with the velocity change due to the computed impulses for a pseudo-time of $\Delta\tau$. The above procedure is iterated over the bodies and is both more accurate and converges faster than an analogous pairwise method.

Equations (4.1) and (4.2) apply generally to both rigid bodies and deformable particles, with the simplification that $\mathbf{r}_o = \mathbf{0}$ for a central deformable particle (reducing Equation (4.2) to a 3×3 system) and that $\mathbf{q}_o = \mathbf{0}$ for an outer deformable particle. When the central body is static or kinematic (having infinite inertia) the equations for the outer bodies decouple as the first two terms in Equation (4.1) vanish. When any outer bodies are static or kinematic the system must be decomposed into the degrees of freedom determined by the static bodies and the remaining degrees of freedom corresponding to the nullspace of the equations for the static bodies. Afterwards, the

system can be reassembled and solved. For more details see Appendix B.

We note that in practice applying arbitrary forces and torques to resolve interpenetration may cause problems. It may be more advisable to apply only normal forces as in Section 4.7.3. In this case, however, the nullspace will need to be computed numerically, which will carry additional cost. Another option is to construct a global system, which could be solved with conjugate gradient. The global solve can then be repeated to convergence. This would avoid the potential for biasing, though we have not observed ill effects from biasing in practice.

4.7 Step V: Advance Velocity

The second half of the Newmark time integration scheme is the velocity update, which we do as follows:

1. Make incompressible $\overline{\mathbf{v}}^n \rightarrow \mathbf{v}_\star^n$
2. $\mathbf{v}_\star^{n+1} = \mathbf{v}_\star^n + \Delta t\, \hat{\mathbf{a}}(t^{n+\frac{1}{2}}, \frac{1}{2}(\mathbf{x}^n + \mathbf{x}^{n+1}), \frac{1}{2}(\mathbf{v}_\star^n + \mathbf{v}_\star^{n+1}))$
3. $\mathbf{v}_{\star\star}^{n+1} = \mathbf{v}_\star^n + \Delta t\, \mathbf{a}(t^{n+\frac{1}{2}}, \frac{1}{2}(\mathbf{x}^n + \mathbf{x}^{n+1}), \frac{1}{2}(\mathbf{v}_\star^n + \mathbf{v}_\star^{n+1}))$
4. Apply constraints: post-stabilization, PD control, contact, post-stabilization and self-repulsions $\mathbf{v}_{\star\star}^{n+1} \rightarrow \mathbf{v}^{n+1}$

We project the velocity field for incompressibility before our velocity evolution as in [80]. In step 2, we use a variant of trapezoid rule to solve for the velocities. In practice, we break this step into a backward Euler solve followed by extrapolation as in [138]. We note that one could instead use backward Euler, which introduces more damping. We discuss steps 2-4 in more detail below.

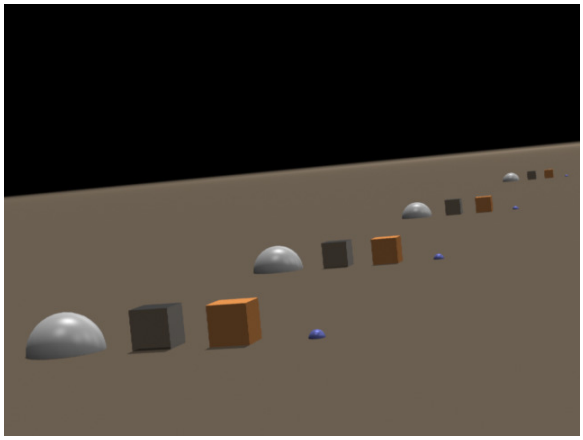


Figure 4.8: Objects slide down an inclined plane with friction. From left to right: analytic solution, rigid box, deformable box and a single particle.

4.7.1 Constrained Solve

The linear system in step 2 is solved using conjugate gradient. We include as many linear constraints as possible in the conjugate gradient solve, accounting for post-stabilization of joints between rigid bodies and two-way contact. This is desirable because it allows the effects of constraints to propagate instantly during the implicit solve. Note that $\hat{\mathbf{a}}$ has been used instead of $\mathbf{a}$ in step 2 above to indicate that these forces are modified during the conjugate gradient solve by applying projections to satisfy constraints as in [12].

4.7.2 Final Velocities

As shown in [12], the unprojected residual of the converged system in step 2 gives the net normal force due to all constraints. [12, 130] use the unprojected residual to compute frictional forces due to contact between particles and a single immovable body. They are able to postprocess each particle independently based on the normal force it feels because they enforce contact in a one-way fashion. That is, computing the frictional forces on a particle results in a change in the particle's tangential velocity relative to the body, but does not affect the body's velocity. Thus, the particle/body

interactions are decoupled, with the body acting as though it had infinite inertia.

In contrast to [12, 130], the interactions we compute are fully two-way. Thus, we cannot postprocess constrained particles for friction independently after the solve, since these constraints are not decoupled. Furthermore, the residual only yields the net force, so we would not be able to untangle the effects of multiple interactions. Therefore, we propose a novel framework of incorporating the constraints into conjugate gradient only for the purpose of determining damping forces (and elastic forces if they are treated implicitly) and apply friction and contact as a postprocess. Step 3 uses the result of step 2 to compute the acceleration $\mathbf{a}$ as opposed to the projected acceleration $\hat{\mathbf{a}}$. Thus, all of the projections in step 2 only help to determine what velocity $\mathbf{v}_\star^{n+1}$ will be used to evaluate the damping forces in step 3. This approach produces accurate friction as illustrated in Figure 4.8.

An issue in our approach is that the postprocess of the velocities for constraint forces does not allow the damping forces to act on forces applied during the postprocess until the next time step. Thus, one needs to realize that there are undamped velocities, such as when computing a CFL condition. In addition, those velocities may get damped based on a smaller time step than the time step in which they were applied. We have nevertheless found that we can work around this by simply moving the postprocesses after applying explicit forces but before the conjugate gradient solve (the postprocesses still see explicit forces but not implicit forces and will thus be less accurate). We note that this was only done for the large pile example, and all other examples were run as described above.

4.7.3 Enforcing Constraints in the Solve

We enforce the linear constraints in the conjugate gradient solve in a momentum-conserving way. Let the constraints be written as $\mathbf{C}^T\mathbf{v} = 0$, where $\mathbf{C}$ is a matrix of coefficients and $\mathbf{v}$ is a vector containing the linear and angular velocities of all particles and rigid bodies. If $\mathbf{M}^{-1}$ is the block-diagonal mass matrix, the projection

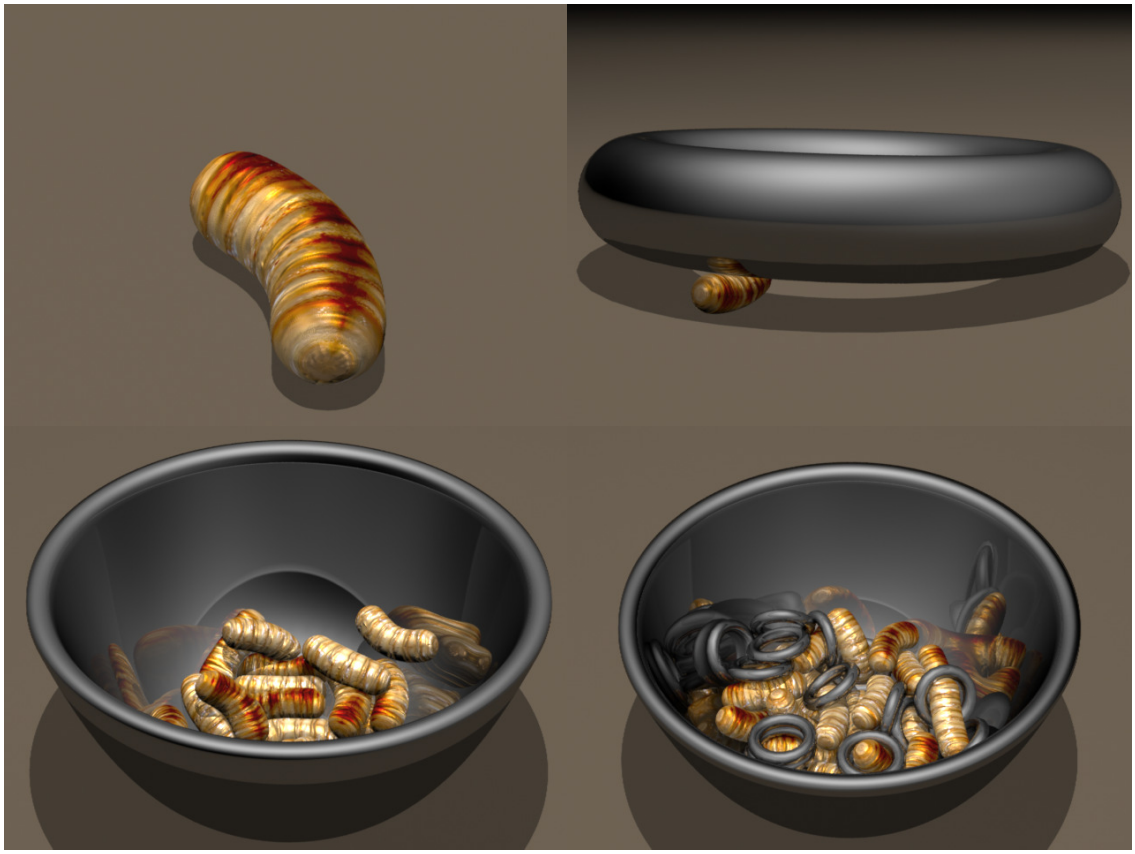


Figure 4.9: We construct maggot by embedding a two-joint PD-controlled articulated rigid body skeleton into a deformable body. The top row shows a maggot on the ground and trapped under a large rigid body. The bottom row shows 20 maggots dropped into a bowl wriggling and interacting with each other, and the last image also has them interacting with 20 rigid tori.

to be applied is

$$\mathbf{P} = \boldsymbol{\delta} - \mathbf{M}^{-1}\mathbf{C}(\mathbf{C}^T\mathbf{M}^{-1}\mathbf{C})^{-1}\mathbf{C}^T,$$

where $\boldsymbol{\delta}$ is the identity matrix. This general form for the projection matrix holds for both rigid bodies and individual particles. When there are multiple constraints, we iterate these projections using forward and backward sweeps to ensure our projection step is symmetric even if it has not fully converged as in Section 3.4 and [80]. The resulting matrix is always symmetric positive semidefinite and can be solved with conjugate gradient provided the residual is properly projected at each iteration.

Figure 4.10: We construct a deformable fish by embedding a four-joint PD-controlled articulated rigid body skeleton inside a deformable fish model. The fish is dropped on the ground and flops back and forth interacting with its environment.

An alternative approach would be to solve the augmented system (or KKT system), which is symmetric and indefinite. This system was solved efficiently in [10] for the case of explicitly integrated applied forces and a loop-free set of constraints. In the case of loops, such as those introduced by loops of articulated rigid bodies or implicitly integrated damping forces, this approach is no longer efficient. Another possible approach is to solve the KKT system with an iterative Krylov solver for indefinite systems, avoiding the projections at the cost of potentially inferior convergence characteristics. This KKT system can also be converted into an SPD system using the SPD framework described in Chapter 5. This formulation is developed in Section 5.10.

Post-stabilization The post-stabilization algorithm of [169] is included in our conjugate gradient solve to project the rigid body velocities in a momentum-conserving fashion so that they satisfy joint constraints. Since step 3 discards these velocities, they are reapplied in step 5.

Rigid/rigid contact We incorporate rigid/rigid contact constraints into the solve by creating joints just prior to the solve and removing them afterwards. In this fashion, we are able to use the articulated rigid body post-stabilization algorithm for projecting contact constraints during the conjugate gradient solve. When computing contact (during step 3 of Section 4.5) we record all points on the surface of one rigid body processed for contact against another. For each such point, we use its time t^{n+1} location and the level set normal from the other body at that point. We then construct a joint that only constrains motion in the normal direction and leaves the other two prismatic degrees of freedom and all angular degrees of freedom unconstrained.

Rigid/deformable contact Rigid/deformable contact projection is performed in a manner similar to interpenetration resolution as described in Section 4.6.1. We process a rigid body against all particles in contact with it simultaneously. However, we restrict impulses to the normal direction $\mathbf{n}_o$ during this contact processing phase. We target a relative change $\mathbf{v}^{n+1}$ that will cancel out the relative velocities in the normal direction of the particle and the body at the intersection point $\mathbf{p}_o$. If the central body is kinematic, the particles are given an impulse in the normal direction that cancels the relative normal velocity between the particle and the rigid body. Otherwise we apply impulses $j_o\mathbf{n}_o$ (where j_o is a scalar) to each outer body c at $\mathbf{p}_o$ so that

$$\mathbf{n}_o^T(m_c^{-1}\mathbf{j} + \mathbf{r}_o^{*T}\mathbf{I}_c^{-1}\mathbf{j}_\tau - \mathbf{K}_o j_o \mathbf{n}_o) = \mathbf{n}_o^T\mathbf{v}^{n+1}. \tag{4.3}$$

Using

$$\mathbf{L}_o = \mathbf{n}_o(\mathbf{n}_o^T\mathbf{K}_o\mathbf{n}_o)^{-1}\mathbf{n}_o^T,$$

we can express these equations in the form

$$\begin{pmatrix} m_c\boldsymbol{\delta} + \sum_o \mathbf{L}_o & \sum_o \mathbf{L}_o \mathbf{r}_o^{*T} \\ \sum_o \mathbf{r}_o^* \mathbf{L}_o & \mathbf{I}_c + \sum_o \mathbf{r}_o^* \mathbf{L}_o \mathbf{r}_o^{*T} \end{pmatrix} \begin{pmatrix} m_c^{-1}\mathbf{j} \\ \mathbf{I}_c^{-1}\mathbf{j}_\tau \end{pmatrix} = \begin{pmatrix} -\sum_o \mathbf{L}_o \mathbf{v}^{n+1} \\ -\sum_o \mathbf{r}_o^* \mathbf{L}_o \mathbf{v}^{n+1} \end{pmatrix}, \tag{4.4}$$

where the net linear and angular impulses applied to the central body are $\mathbf{j} = -\sum_o j_o \mathbf{n}_o$ and $\mathbf{j}_\tau = -\sum_o \mathbf{r}_o^* j_o \mathbf{n}_o$. This 6×6 system is symmetric positive definite and can be solved to obtain $\mathbf{j}$ and $\mathbf{j}_\tau$, from which j_o is obtained by substitution into Equation (4.3).

Self-repulsions Self-repulsions apply forces between edge-edge and point-triangle pairs to help avoid collisions [27]. In the conjugate gradient solve, we use the projection algorithm proposed in [80] to enforce these constraints.

4.8 Examples

Figure 4.5 illustrates that we can use the embeddings of [138] in our fully two-way coupled approach. Note that [138] was unable to handle two-way collisions and contact and more generally used an interleaved, semi-implicit approach to rigid/deformable coupling. Figure 4.1 demonstrates the robustness of the algorithm which allows stable two-way interaction between many types of competing constraints. Figure 4.4 demonstrates the ability of the two-way contact and collision algorithms to handle both volumetric objects and thin shells. Figure 4.6 demonstrates the ability of the algorithm to handle simulations with large numbers of two-way coupled bodies. One of the major applications of two-way rigid/deformable coupling is the simulation of skeleton-controlled deformable objects that can interact with their environment as shown in Figures 4.7, 4.9 and 4.10. Table 4.1 provides timing information for all of the examples in this paper. Most of the examples were fast enough that we simply ran them with conservative time steps, whereas the large pile was expensive enough to warrant performance tuning.

	Ave. Time Per Frame	Ave. Substeps Per Frame
Twisting Chain (Figure 4.1)	92.2 sec†	123
Stack (Figure 4.2)	10.0 sec	35
Row of Spheres (Figure 4.3)	35.3 sec	94
Deformable Bar (Figure 4.5)	4.1 sec	51
Trampoline (Figure 4.4)	14.5 sec	32
Large Pile (Figure 4.6)	40 min	14
Snake (Figure 4.7)	1.8 sec	38
Friction (Figure 4.8)	6.4 sec	2515^{+}
Single Maggot (Figure 4.9)	0.6 sec	30
Maggot and Ring (Figure 4.9)	7.7 sec	100
Bowl of Maggots (Figure 4.9)	19.1 sec	30
... with Rings (Figure 4.9)	62.2 sec	59
Fish (Figure 4.10)	52.8 sec	117

Table 4.1: This table contains average times per frame and number of substeps required per frame for each example. † The individual times varied substantially between frames, depending strongly on the degree of stress. $^{+}$ This test was run as a convergence test with an artificially low time step.

4.9 Conclusions

We propose a novel time integration scheme for two-way coupling rigid and deformable bodies that retains the strengths of deformable object simulators and rigid body simulators. We build upon existing algorithms for rigid/rigid and deformable/deformable interaction and where necessary propose new fully coupled algorithms. The resulting scheme handles two-way coupled contact, collision, stacking, friction, articulation, and PD control. We use our framework to simulate life-like creatures that interact with their environment.

Chapter 5

Fluid Structure Interaction

The previous chapter addressed monolithic coupling of rigid bodies and deformable bodies. In this chapter we consider a strongly coupled (monolithic) fluid structure interaction framework for incompressible flow. We take a novel approach that consists of factoring the damping matrix of deformable structures and show that this can be used to obtain a symmetric positive definite system. The approach extends readily to include fluid viscosity and arbitrary linear constraints on the fluid or structure velocities. Under some circumstances, the same manipulations can be used to convert a symmetric but indefinite KKT system into one that is SPD. For the special case of rigid bodies, where there are no internal damping forces, we recover the system of [15]. This chapter is based on the work published in [124].

5.1 Introduction

Fluid structure interaction has been of increasing interest to the computational fluids community, due in part to the ability to address more computationally expensive problems with improved hardware. Researchers have found differing approaches useful for qualitatively different regimes of fluid structure interaction. Applications may

involve low or high structural deformation. Problems involving low deformation can be approached with aligning fluid meshes using for instance Arbitrary Lagrangian-Eulerian (ALE) schemes. This has been quite successful for studying phenomena such as airplane wing flutter, see e.g. [51]. High structural deformation is sometimes also approached using ALE techniques, but frequent remeshing and the averaging it requires can lead to reduced accuracy and increased computational cost. More often, methods where the fluid and structure meshes are not aligned are employed (e.g. the body of work originating from [117]). In this case the interaction between fluid and structure is usually massaged via penalty methods or Lagrange multiplier formulations.

Anecdotal evidence suggests that problems where either the fluid or the structure dominates the interaction are inherently more stable than those where the effect is more balanced. For example, it is more straightforward to solve problems where a kinematic structure drives the fluid or a structure is passively advected by fluid, as opposed to problems where structures and fluids have nearly equal densities (see e.g. [30]). Partitioned coupling approaches are typically explicit and staggered, for instance first imposing the fluid pressures as a force on the structure and then solving for the fluid using the structure velocities as boundary conditions in a second step. These approaches can also be iterated within a given time step for increased stability, noting that one obtains a monolithic (albeit expensive) approach if the iteration is run to convergence. Other approaches construct strongly coupled systems and then solve them in one of several ways. We follow a monolithically coupled approach, in which we model the fluid structure interaction by formulating and solving a single coupled system of equations; however, we do this in a way which is more efficient as well as more likely to converge than simply iterating the partitioned approach to convergence in a simple Gauss-Seidel manner.

Our approach allows the use of existing discretizations for both the fluid and the structure and requires only minor modification to existing codes. We do require that the fluid and structure solvers have certain properties, though we will argue that they ought to be true of any physically constituted system. The current formulation

assumes that the fluid and structure systems are synchronized in time such that we can solve for the coupling at the synchronization points.

The method that we propose in this paper is similar to the previous work in [125] addressing an implicit method for two-way coupling of incompressible fluids and structures, with the chief difference being that the linear system presented in [125] was indefinite. Our main contribution is the observation in Section 5.5.2 that the structure damping operator can be factored and used to perform a change of variables that allows us to symmetrize the coupled system while maintaining positive definiteness. We examine the properties of the resulting linear system and demonstrate convergence of the method on several example problems from the literature.

5.2 Fluid Equations

We address the Navier-Stokes equations for incompressible flow,

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}\right) = -\nabla p + \mu\nabla^2\mathbf{u} + \mathbf{f} \tag{5.1}$$

and

$$\nabla\cdot\mathbf{u} = 0, \tag{5.2}$$

in the presence of structures. Here, $\mathbf{u}$ is the fluid velocity, p is the pressure, μ is the dynamic viscosity, ρ is the density, and $\mathbf{f}$ is any external body force such as gravity.

We use a standard Marker and Cell (MAC, [68]) grid discretization for the fluid. When solving in the presence of structures, some faces may not contain valid velocity samples because they are covered by structures. The vector of fluid velocity samples omitting any covered by the structure is denoted by $\mathbf{u}$. Pressure samples are defined at cell centers. The divergence operator is defined to act on velocities at cell faces to yield divergences at cell centers using the standard six-point stencil in three spatial

dimensions, and the gradient operator is defined as the negative transpose of the divergence.

We apply the projection method of [37] to solve the discretized form of (5.1) and (5.2), making a first order approximation in time. This splits the time integration into the computation of an intermediate velocity ignoring the pressure terms

$$\mathbf{u}^\star = \mathbf{u}^n - \Delta t(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n + \frac{\Delta t \mu}{\rho}\nabla^2\mathbf{u} + \frac{\Delta t}{\rho}\mathbf{f} \tag{5.3}$$

followed by enforcement of incompressibility via an implicit pressure projection

$$\mathbf{u}^{n+1} = \mathbf{u}^\star - \frac{\Delta t}{\rho}\nabla p. \tag{5.4}$$

We have omitted a specification of time from the viscosity term, since the viscosity might be applied explicitly or implicitly. We postpone further consideration of viscosity until Section 5.7, at which point we will include it in the coupling framework. We use a semi-Lagrangian advection scheme (see e.g. [129]) to compute the convective terms in (5.3). In the presence of structures, this requires ghost cells or clipping since the back-cast rays may intersect the solid; we use a method based on [65] for semi-Lagrangian advection near solid surfaces. We advect using the divergence-free time n velocities. We enforce incompressibility as usual by plugging (5.4) into the discrete form of (5.2) to obtain

$$\nabla \cdot \left(\frac{\Delta t}{\rho}\nabla p\right) = \nabla \cdot \mathbf{u}^\star,$$

which we solve for p using the conjugate gradient method.

5.3 Structure Equations

We utilize a traditional Lagrangian approach for structures. For deformable structures, we solve the equations $\dot{\mathbf{x}} = \mathbf{v}$ and $\mathbf{M}\dot{\mathbf{v}} = \mathbf{F}$. We take a lumped mass approach

where the quantities $\mathbf{x}$ and $\mathbf{v}$ are the vectors containing the position and velocity degrees of freedom for all n of the particles. The mass matrix $\mathbf{M}$ is a $3n \times 3n$ diagonal matrix with each 3×3 diagonal block of the form $m_i\mathbf{I}$, where m_i is the mass of particle i. The forces $\mathbf{F} = \mathbf{F}_e + \mathbf{F}_d$ contain both elastic $\mathbf{F}_e$ (velocity independent) and damping $\mathbf{F}_d$ (velocity dependent) forces. Purely elastic forces (e.g., no plasticity) can be expressed in terms of an energy potential as $\mathbf{F}_e = -\frac{\partial \phi}{\partial x}$, where the energy potential ϕ depends only on $\mathbf{x}$. Damping forces are often chosen according to a simple model $\mathbf{F}_d = \mathbf{D}\mathbf{v}$ that is linear in velocities and has symmetric negative semidefinite $\mathbf{D}$. This is true, for example, for the damping applied to a simple spring and Rayleigh damping.

One particularly simple and stable way of evolving these equations is via backward Euler. We wish to solve

$$\begin{pmatrix} \mathbf{x}^{n+1} \\ \mathbf{v}^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{x}^n + \Delta t \mathbf{v}^{n+1} \\ \mathbf{v}^n + \Delta t \mathbf{M}^{-1}\mathbf{F}^{n+1} \end{pmatrix}.$$

Taking the first order approximation $\mathbf{F}^{n+1} = \mathbf{F}^n + \frac{\partial \mathbf{F}^n}{\partial \mathbf{x}}(\mathbf{x}^{n+1} - \mathbf{x}^n) + \frac{\partial \mathbf{F}^n}{\partial \mathbf{v}}(\mathbf{v}^{n+1} - \mathbf{v}^n)$ this becomes

$$\begin{pmatrix} \mathbf{x}^{n+1} \\ \mathbf{v}^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{x}^n + \Delta t \mathbf{v}^{n+1} \\ \mathbf{v}^n + \Delta t \mathbf{M}^{-1}(\mathbf{F}^n + \frac{\partial \mathbf{F}^n}{\partial \mathbf{x}}(\mathbf{x}^{n+1} - \mathbf{x}^n) + \frac{\partial \mathbf{F}^n}{\partial \mathbf{v}}(\mathbf{v}^{n+1} - \mathbf{v}^n)) \end{pmatrix}.$$

Using the first equation to eliminate $\mathbf{x}^{n+1}$ in the second equation yields

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta t \mathbf{M}^{-1}(\mathbf{F}^n + \frac{\partial \mathbf{F}^n}{\partial \mathbf{x}}\Delta t \mathbf{v}^{n+1} + \frac{\partial \mathbf{F}^n}{\partial \mathbf{v}}(\mathbf{v}^{n+1} - \mathbf{v}^n))$$

$$\left(\mathbf{I} - \Delta t \mathbf{M}^{-1}\frac{\partial \mathbf{F}^n}{\partial \mathbf{v}} - \Delta t^2 \mathbf{M}^{-1}\frac{\partial \mathbf{F}^n}{\partial \mathbf{x}}\right)\mathbf{v}^{n+1} = \mathbf{v}^n - \Delta t \mathbf{M}^{-1}\frac{\partial \mathbf{F}^n}{\partial \mathbf{v}}\mathbf{v}^n + \Delta t \mathbf{M}^{-1}\mathbf{F}^n \quad (5.5)$$

From this we see that we must invert $\mathbf{I} - \Delta t \mathbf{M}^{-1}\frac{\partial \mathbf{F}^n}{\partial \mathbf{v}} - \Delta t^2 \mathbf{M}^{-1}\frac{\partial \mathbf{F}^n}{\partial \mathbf{x}}$. The second term, $-\Delta t \mathbf{M}^{-1}\frac{\partial \mathbf{F}}{\partial \mathbf{v}} = -\Delta t \mathbf{M}^{-1}\mathbf{D}$, is symmetric positive semidefinite under a mass weighted inner product, which can be used for solving the equation. The third term, however, can cause problems. Note that $\frac{\partial \mathbf{F}}{\partial \mathbf{x}} = \frac{\partial \mathbf{F}_e}{\partial \mathbf{x}} + \frac{\partial \mathbf{F}_d}{\partial \mathbf{x}}$. The first term is symmetric since

$\mathbf{F}_e = -\frac{\partial \phi}{\partial x}$, but it might be indefinite. The second term need not even be symmetric under the mass weighted inner product.

Another approach is a Newmark-style semi-implicit scheme

- $\mathbf{v}_\star^{n+1/2} = \mathbf{v}^n + \frac{\Delta t}{2}\mathbf{M}^{-1}\mathbf{F}(\mathbf{x}^n, \mathbf{v}_\star^{n+1/2})$
- $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}_\star^{n+1/2}$
- $\mathbf{v}^{n+1/2} = \mathbf{v}^n + \frac{\Delta t}{2}\mathbf{M}^{-1}\mathbf{F}(\mathbf{x}^n, \mathbf{v}^n)$
- $\mathbf{v}^{n+1} = \mathbf{v}^{n+1/2} + \frac{\Delta t}{2}\mathbf{M}^{-1}\mathbf{F}(\mathbf{x}^{n+1}, \mathbf{v}^{n+1})$

Each time step of this scheme involves two implicit solves, bulleted steps 1 and 4. These solves differ from the fully implicit backward Euler solve in that the dependence of force on position is explicit. The first solve, if taken for a full Δt, would look like

$$\begin{aligned}
\mathbf{v}^{n+1} &= \mathbf{v}^n + \Delta t \mathbf{M}^{-1}\mathbf{F}(\mathbf{x}^n, \mathbf{v}^{n+1}) \\
\mathbf{v}^{n+1} &= \mathbf{v}^n + \Delta t \mathbf{M}^{-1}(\mathbf{F}^n + \frac{\partial \mathbf{F}^n}{\partial \mathbf{v}}(\mathbf{v}^{n+1} - \mathbf{v}^n)) \\
\left(\mathbf{I} - \Delta t \mathbf{M}^{-1}\frac{\partial \mathbf{F}^n}{\partial \mathbf{v}}\right)\mathbf{v}^{n+1} &= \mathbf{v}^n - \Delta t \mathbf{M}^{-1}\frac{\partial \mathbf{F}^n}{\partial \mathbf{v}}\mathbf{v}^n + \Delta t \mathbf{M}^{-1}\mathbf{F}^n \qquad (5.6) \\
\big(\mathbf{I} - \Delta t \mathbf{M}^{-1}\mathbf{D}\big)\mathbf{v}^{n+1} &= \mathbf{v}^n - \Delta t \mathbf{M}^{-1}\mathbf{D}\mathbf{v}^n + \Delta t \mathbf{M}^{-1}\mathbf{F}^n \qquad (5.7)
\end{aligned}$$

We see from this that the fully implicit system (5.5) differs from the semi-implicit system (5.6) by the presence of the extra $-\Delta t^2 \mathbf{M}^{-1}\frac{\partial \mathbf{F}^n}{\partial \mathbf{x}}$ term. This term was the source of the indefiniteness and asymmetry problems in the fully coupled system. This simplification comes at the price of stability, however. The fully implicit backward Euler scheme is unconditionally stable, but the semi-implicit scheme is not. Of course, this is the well-known difference between a fully implicit Newmark method and one that resembles central differencing [76].

The particular scheme used for the simulations in this paper is based on the backward Euler variant of [134], which differs from the previous scheme in that it has been rewritten so that a backward Euler step can be used for updating the final positions. That is

1. $\hat{\mathbf{v}}^\star = \mathbf{v}^n + \frac{\Delta t}{2}\mathbf{M}^{-1}\mathbf{F}_e(\mathbf{x}^n)$
2. $\mathbf{v}^{n+1/2} = \hat{\mathbf{v}}^\star + \frac{\Delta t}{2}\mathbf{M}^{-1}\mathbf{D}(\mathbf{x}^n)\mathbf{v}^{n+1/2}$
3. $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t\mathbf{v}^{n+1/2}$
4. $\mathbf{v}^\star = \mathbf{v}^n + \Delta t\mathbf{M}^{-1}\mathbf{F}_e(\mathbf{x}^{n+1})$
5. $\mathbf{v}^{n+1} = \mathbf{v}^\star + \Delta t\mathbf{M}^{-1}\mathbf{D}(\mathbf{x}^{n+1})\mathbf{v}^{n+1}$

As before, both solves lead to systems of the form (5.7). This scheme is first order accurate in time, limited to first order by the backward Euler step used to compute the final velocities.

Our treatment of rigid bodies follows that of [134]. In general, each particle of a deformable body contributes one block to $\mathbf{v}$, $\mathbf{M}$, and $\mathbf{F}$. Each rigid body contributes two blocks to each, with the first block corresponding to the linear part, $\mathbf{v}_i$, m_i, and $\mathbf{f}_i$, and the second block corresponding to the angular part, $\boldsymbol{\omega}_i$, $\mathbf{I}_i$, and $\boldsymbol{\tau}_i$. With this taken into account, the linear system (5.7) is valid for deformable and rigid bodies.

The ability to treat rigid bodies using the same formulas as deformable bodies breaks down somewhat when dealing with advancing rigid body orientation. Fortunately, the coupling formulation presented here is explicit in position and therefore does not directly involve the integration of rigid body orientation, so this complication is avoided. The coupling formulation is valid when bodies are deformable, rigid, or a mixture of both.

5.4 Discrete Fluid Structure Coupling Equations

Our goal is to write down the equations that need to be solved in full, including the coupling and any other terms such as boundary conditions. In doing this it is more useful to work from the actual discretizations than it is from the continuous equations. Therefore we will consider the MAC grid fluid discretizations and the Lagrangian

structure discretizations as defined in Sections 5.2 and 5.3 and use them going forward for our notation. However, we stress that one could use other discretizations for the fluids and structures and apply the same methodology that we propose to those sets of equations.

We follow a method similar to that of [125] to merge the time integration schemes described in Sections 5.2 and 5.3,

1. $\hat{\mathbf{v}}^\star = \mathbf{v}^n + \frac{\Delta t}{2}\mathbf{M}^{-1}\mathbf{F}_e(\mathbf{x}^n)$
2. $\hat{\mathbf{u}}^\star = \mathbf{u}^n + \frac{\Delta t}{2\rho}\mathbf{f}$
3. Coupled solve with known quantities $\hat{\mathbf{u}}^\star$, $\hat{\mathbf{v}}^\star$, and $\mathbf{x}^n$ and unknown quantities $\mathbf{u}^{n+1/2}$, $\mathbf{p}$, and $\mathbf{v}^{n+1/2}$
4. $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t\mathbf{v}^{n+1/2}$
5. $\mathbf{v}^\star = \mathbf{v}^n + \Delta t\mathbf{M}^{-1}\mathbf{F}_e(\mathbf{x}^{n+1})$
6. $\mathbf{u}^\star = \mathbf{u}^n - \Delta t(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n + \frac{\Delta t}{\rho}\mathbf{f}$
7. Coupled solve with known quantities $\mathbf{u}^\star$, $\mathbf{v}^\star$, and $\mathbf{x}^{n+1}$ and unknown quantities $\mathbf{u}^{n+1}$, $\mathbf{p}$, and $\mathbf{v}^{n+1}$

Steps 1 through 2 integrate the explicit structure forces and explicit fluid forces for $\Delta t/2$. We leave out the convective terms in the fluid update because the coupled solve in step 3 takes positions at time n and solves for velocities at time $n + \frac{1}{2}$. The known quantities at the beginning of step 3, which replaces step 2 in Section 5.3, are $\hat{\mathbf{u}}^\star$, $\hat{\mathbf{v}}^\star$, and $\mathbf{x}^n$. The remaining unknown quantities to solve for in (5.2), (5.4) and (5.7) are $\mathbf{u}^{n+1/2}$, $\mathbf{p}$, and $\mathbf{v}^{n+1/2}$. The details of the coupled solve are described later, but note that this solve takes place over only half of the time step (and thus has step size $\frac{\Delta t}{2}$). In step 4, $\mathbf{v}^{n+1/2}$ is used to advance the structure positions to time $n + 1$, and then $\mathbf{u}^{n+1/2}$, $\mathbf{p}$ and $\mathbf{v}^{n+1/2}$ are discarded.

Steps 5 through 6 re-integrate the explicit structure forces and explicit fluid forces for Δt, this time including the convective terms in the fluid update to get time $n + 1$

fluid positions, resulting in $\mathbf{v}^\star$ and $\mathbf{u}^\star$. The known quantities at the beginning of step 7, which replaces step 5 from Section 5.3 and (5.4), are $\mathbf{u}^\star$, $\mathbf{v}^\star$, and $\mathbf{x}^{n+1}$. We use $\mathbf{x}^{n+1}$ to determine the valid degrees of freedom for $\mathbf{u}^{n+1}$. The remaining unknown quantities to solve for in (5.2), (5.4) and (5.7) are $\mathbf{u}^{n+1}$, $\mathbf{p}$, and $\mathbf{v}^{n+1}$. The coupled solve used in this step has the same form as that in step 3.

For the sake of exposition, we consider below only the second of the two implicit steps described above (step 7), and therefore we will discuss the updates from $\mathbf{u}^\star$ and $\mathbf{v}^\star$ to $\mathbf{u}^{n+1}$ and $\mathbf{v}^{n+1}$, whereas the updates from $\hat{\mathbf{u}}^\star$ and $\hat{\mathbf{v}}^\star$ to $\mathbf{u}^{n+1/2}$ and $\mathbf{v}^{n+1/2}$ will be handled independently. One can replace Δt, $\mathbf{x}^{n+1}$, $\mathbf{u}^\star$, $\mathbf{v}^\star$, $\mathbf{u}^{n+1}$ and $\mathbf{v}^{n+1}$ with $\frac{\Delta t}{2}$, $\mathbf{x}^n$, $\hat{\mathbf{u}}^\star$, $\hat{\mathbf{v}}^\star$, $\mathbf{u}^{n+1/2}$ and $\mathbf{v}^{n+1/2}$ in the discussion that follows in order to apply the same treatment to step 3.

5.4.1 Fluid Structure Coupling Constraints

The modifications to the fluid and structure equations correspond to constraints that capture the behavior at the fluid structure interface. A fluid structure coupling constraint is an enforced constraint on the relative velocity of the fluid and the structure at a point in space. In our implementation, a constraint occurs when a ray cast between two adjacent MAC grid cell centers, at least one of which is in the fluid region, intersects a structure surface. The constraint is placed at the center of the MAC grid face between the two cell centers and enforces that the projection of the fluid and structure velocities at that point in the ray direction are equal. The impulse transfer between the solid and the fluid which enforces the constraint is called $\boldsymbol{\lambda}$. One may also interpret this coupling impulse $\boldsymbol{\lambda}$ as the pressure the fluid applies to the structure, although we note this analogy is only accurate up to a scale. See Figure 5.1. This simple choice of constraint location is only first order accurate, since the constraint locations are located $O(\Delta x)$ from the structure surface.

5.4.2 Modified Fluid Equations

For fluid structure interaction, (5.4) changes due to the interaction with the structure. In Figure 5.1 for the x direction we have a mass in that dual cell m and a velocity u, the product of which mu is equal to $mu^\star$ plus the contribution from p in the fluid on the left side of the dual cell, plus a contribution from λ due to interaction with the structure. This discards the pressure on the right hand side of the dual cell due to the structure. We can neglect the dual cells that would lose two pressure contributions, those inside the structure, since they are not degrees of freedom. We write this in vector form for all of the fluid velocity terms as

$$\boldsymbol{\beta}\mathbf{u}^{n+1} = \boldsymbol{\beta}\mathbf{u}^\star - \mathbf{G}\hat{\mathbf{p}} + \mathbf{W}^T\boldsymbol{\lambda}, \tag{5.8}$$

where $\boldsymbol{\beta}$ is the diagonal matrix of fluid dual cell masses assembled from the product of the density ρ and the volume V of each dual cell. We let $\hat{\mathbf{p}} = \Delta t\mathbf{p}$. We define $-\mathbf{G}^T = -V\nabla^T$, the volume weighted divergence, with the additional modification that we drop rows from $-\mathbf{G}^T$ that correspond to cells without fluid. Its negative transpose, $\mathbf{G}$, is defined as the volume weighted gradient. $-\mathbf{G}^T$ is simpler to describe directly than $\mathbf{G}$, since every row of $-\mathbf{G}^T$ has the same stencil while the stencils for rows of $\mathbf{G}$ differ because some pressure samples fall inside the structure. $\mathbf{W}^T$ is the matrix of 1s and 0s which maps $\boldsymbol{\lambda}$ to the appropriate fluid velocity scalars, which will be discussed in more detail below. The discrete form of the incompressibility constraint is

$$-\mathbf{G}^T\mathbf{u}^{n+1} = \mathbf{0}. \tag{5.9}$$

Our discretizations of $\mathbf{W}$, $\mathbf{G}$, and $-\mathbf{G}^T$ are first order accurate since the constraint locations used in their definition are only first order accurate. We note that even were $\mathbf{G}$ a higher order accurate approximation of the gradient operator it would not necessarily imply that $-\mathbf{G}^T$ would be a compatible and higher order accurate approximation of the divergence. Unlike the second order discretizations [57, 108], it

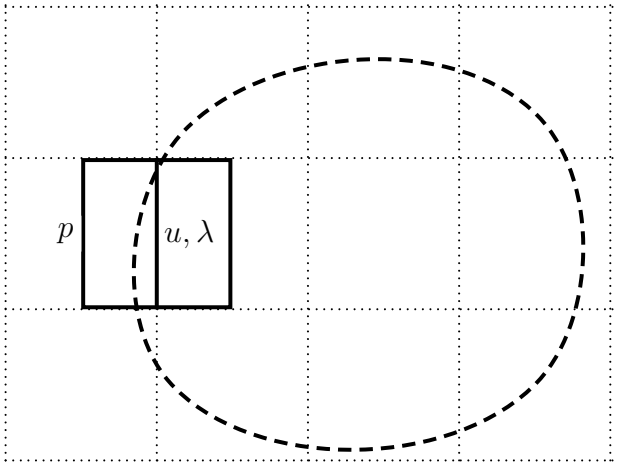


Figure 5.1: A dual cell requiring an equal velocity constraint is depicted above intersecting the solid depicted by the dashed line. The fluid pressure on the left face of the dual cell will be balanced by the constraint-maintaining impulse λ applied at the equal velocity constraint location.

is not sufficient in this context to discretize the Poisson operator with higher order accuracy. The discretizations of gradient and divergence in [108] do not satisfy the negative transpose relationship, so using them in this formulation would produce an asymmetric linear system.

5.4.3 Modified Structure Equations

To write the modified equations for the structure, one needs to consider the fact that the fluid and structure degrees of freedom are defined at different spatial locations. This is quite important as constraints must be applied to co-located fluid and structure velocities, and the equal and opposite forces applied to the fluid and the structure must be applied at the same location in space so as not to introduce a net torque. We define an operator $\mathbf{J}$ which interpolates from structure degrees of freedom to fluid velocity locations. Its transpose $\mathbf{J}^T$ is a conservative redistribution from fluid velocity locations to structure velocity degrees of freedom.

$\mathbf{J}$ can be decomposed into three matrices, $\mathbf{J} = \mathbf{N}\hat{\mathbf{J}}\mathbf{R}$. The matrix $\mathbf{R}$ is comprised of

blocks $\mathbf{R}_{pb}$, where body b is sampled by particle p. For particles within a deforming body, the particles are the samples, and $\mathbf{R}_{pb} = \mathbf{I}$. If b corresponds to a rigid body, then there is a sample point p at each nearby fluid velocity location. If such a sample point p is displaced from the rigid body's center of mass by $\mathbf{r}_{pb}$, the velocity at the sample point is

$$\mathbf{v}_p = \mathbf{v}_b + \mathbf{r}_{pb}^{*T}\boldsymbol{\omega}_b = \begin{pmatrix} \mathbf{I} & \mathbf{r}_{pb}^{*T} \end{pmatrix} \begin{pmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{pmatrix}_b = \mathbf{R}_{pb} \begin{pmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{pmatrix}_b \tag{5.10}$$

so that $\mathbf{R}_{pb} = \begin{pmatrix} \mathbf{I} & \mathbf{r}_{pb}^{*T} \end{pmatrix}$, where $\mathbf{r}^*$ represents the matrix such that $\mathbf{r}^*\mathbf{u} = \mathbf{r} \times \mathbf{u}$ for any $\mathbf{u}$.

The operator $\hat{\mathbf{J}}$ interpolates particle velocity samples p to a MAC grid face f as $\sum_p \alpha_{fp}\mathbf{v}_p$, so that the blocks of $\hat{\mathbf{J}}$ are $\hat{\mathbf{J}}_{fp} = \alpha_{fp}\mathbf{I}$. Since $\hat{\mathbf{J}}$ is an interpolation operator, $\sum_p \alpha_{fp} = 1$. The coefficients α_{fp} are defined based on local geometric information associated with the face f and the particle p such as the control volume around f and structure faces incident on p. For the rigid body, $\mathbf{R}$ is used to sample the rigid body velocities at sample points, and the most logical positions for sample points are the fluid velocity locations at the centers of faces. In that sense, if only one rigid body was present near a face, locally the $\hat{\mathbf{J}}$ operator is the identity for that degree of freedom. However multiple rigid bodies or particles may influence the same face, in which case an averaging of the velocities would be required, making $\hat{\mathbf{J}}$ nontrivial.

Finally, $\mathbf{N}$ extracts the appropriate scalar velocity component from the vector velocity constructed via $\mathbf{R}$ and $\hat{\mathbf{J}}$. The matrix $\mathbf{N}$ is block diagonal, where each block $\mathbf{N}_{ff}$ corresponds to a grid face f with normal $\mathbf{n}_f$. These blocks are formed as $\mathbf{N}_{ff} = \mathbf{n}_f^T$. Our discretization of the operator $\mathbf{J}$ is limited to first order accuracy since it depends on the first order accurate structure interface location.

Since (5.8) applies an impulse to the fluid at MAC grid face locations, the equal and opposite impulse must be applied to the structure at those same locations, guaranteeing conservation of angular momentum. For the case of rigid bodies, one can apply impulses to the rigid body from any location in space and therefore the method does

conserve angular momentum. However, in the case of deformable bodies $\hat{\mathbf{J}}^T$ contains a spatial redistribution of the impulse which, while being a fully conservative remapping with columns summing to 1 and thus conserving linear momentum, will not conserve angular momentum. In summary, $\mathbf{N}^T$ maps impulse magnitudes from dual cells to impulse vectors normal to the face. Since $\sum_p \alpha_{fp} = 1$, $\hat{\mathbf{J}}^T$ conservatively distributes these impulses to different spatial sample locations in the case of deformable bodies. For rigid bodies the spatial location does not change although the impulse can still be conservatively distributed among multiple rigid bodies influencing that face. Finally $\mathbf{R}^T$ conservatively applies those sample impulses to the structure degrees of freedom. This mapping is trivial for deformable particles, but for rigid bodies it converts the impulse sample into linear and angular impulses at the body's center of mass. We note that [4] proposed a method which allows the solid and fluid domains to be non-coincident, coupled via interpolation and Lagrange multipliers, similar to our use of $\mathbf{W}$ and $\mathbf{J}$.

The full modified structure equations are

$$\mathbf{M}\mathbf{v}^{n+1} = \mathbf{M}\mathbf{v}^{\star} + \Delta t\, \mathbf{D}\mathbf{v}^{n+1} - \mathbf{J}^T\mathbf{W}^T\boldsymbol{\lambda} \tag{5.11}$$

where $\mathbf{M}$ is the structure mass matrix and $\mathbf{D}\mathbf{v}^{n+1}$ is the implicit contribution of structure forces. The last term is the constraint impulse mapped first by $\mathbf{W}^T$ from constraint locations to fluid velocity degrees of freedom and then by $\mathbf{J}^T$ to structure velocity degrees of freedom. The coupling momentum $\mathbf{W}^T\boldsymbol{\lambda}$ is subtracted from the structure equation since it was added to the fluids equation. $\mathbf{W}$ has a number of columns equal to the number of fluid scalar velocity samples and a number of rows equal to the number of constraints. $\mathbf{W}^T$ therefore maps to every MAC grid face but puts zero on every face which is not coupled. As a consequence, $\mathbf{J}^T$ only needs to be nonzero for the faces which are coupled, even though it has a number of columns equal to the number of fluid scalar velocity samples. $\mathbf{J}^T\mathbf{W}^T$ then maps from the fluid degrees of freedom that are coupled to the structure degrees of freedom that are coupled.

Although (5.8) and (5.11) indicate how momentum is conservatively transferred between the fluid and the structure, this still leaves open the question of how much momentum is transferred. That is, we have introduced a new degree of freedom $\boldsymbol{\lambda}$ and have not yet provided the equations to govern these new degrees of freedom. In order to do this we consider the equal velocity constraints that the fluid and the structure should move with the same velocity, or for inviscid flow the same velocity in the normal direction. One way of governing these degrees of freedom is to set the difference between the fluid velocity and the interpolated structure velocity equal to zero at every fluid velocity degree of freedom as indicated in Figure 5.1. This can be expressed as the equation

$$\mathbf{W}(\mathbf{u} - \mathbf{J}\mathbf{v}) = 0, \tag{5.12}$$

noting that $\mathbf{J}$ maps zero to all the face locations that are not influenced by the structure and that $\mathbf{W}$ filters all of these locations out. Note that one can use (5.12) and its variants in order to model a wide range of physical phenomena. For example, one could use $\mathbf{W}(\mathbf{u} - \mathbf{J}(\mathbf{v} - \mathbf{s})) = \mathbf{0}$ in order to model an arbitrary source term with inflow and outflow from the structure. One might also wish to constrain the velocities at other locations than the faces of the MAC grid or to constraint limited components such as the normal component as was done in [123] in order to achieve better slip boundary conditions, but see also [1] for a more thorough discussion of the applicability of slip boundary conditions. In this sense $\mathbf{W}$ may not be simply defined as a trivial restriction matrix and may instead be thought of as a general interpolation operator mapping from the faces of the MAC grid to constraint locations. This would not change the inputs of $\mathbf{W}$, but the outputs would no longer necessarily correspond to the MAC grid faces but could instead be at arbitrary locations. Note that the formulation of our equations does not rule out any of these approaches, but for the sake of exposition we will deal with (5.12) going forward.

5.5 Numerical Approach

We can write our system in symmetric form as

$$\begin{pmatrix} \mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{G} & -\mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{W}^T & \mathbf{0} \\ -\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G} & \mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{W}^T & -\mathbf{W}\mathbf{J} \\ \mathbf{0} & -\mathbf{J}^T\mathbf{W}^T & -\mathbf{M}+\Delta t\,\mathbf{D} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{p}} \\ \boldsymbol{\lambda} \\ \mathbf{v}^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{G}^T\mathbf{u}^\star \\ -\mathbf{W}\mathbf{u}^\star \\ -\mathbf{M}\mathbf{v}^\star \end{pmatrix}.$$

The first equation,

$$\mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{G}\hat{\mathbf{p}} - \mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{W}^T\boldsymbol{\lambda} = \mathbf{G}^T\mathbf{u}^\star \tag{5.13}$$

is obtained by substituting (5.8) into (5.9). The second equation

$$-\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G}\hat{\mathbf{p}} + \mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{W}^T\boldsymbol{\lambda} - \mathbf{W}\mathbf{J}\mathbf{v}^{n+1} = -\mathbf{W}\mathbf{u}^\star \tag{5.14}$$

is obtained by substituting (5.8) into (5.12). The third equation

$$-\mathbf{J}^T\mathbf{W}^T\boldsymbol{\lambda} - \mathbf{M}\mathbf{v}^{n+1} + \Delta t\,\mathbf{D}\mathbf{v}^{n+1} = -\mathbf{M}\mathbf{v}^\star \tag{5.15}$$

is a rearrangement of (5.11).

If we make some assumptions about $\mathbf{W}$ and $\mathbf{J}$, we can eliminate $\boldsymbol{\lambda}$. If $\mathbf{W}$ takes on the particularly simple form described in Sections 5.4.2 and 5.4.3, then $\mathbf{W}\mathbf{W}^T = \mathbf{I}$. If $\mathbf{J}$ has nonzero rows only for constraint faces, then $\mathbf{W}^T\mathbf{W}\mathbf{J} = \mathbf{J}$. If we also take densities to be constant at all constraint faces, then $\mathbf{W}^T\mathbf{W}\boldsymbol{\beta}^{-1} = \boldsymbol{\beta}^{-1}\mathbf{W}^T\mathbf{W}$. Multiplying (5.14) by $\mathbf{W}^T$ produces

$$\begin{aligned} -\mathbf{W}^T\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G}\hat{\mathbf{p}} + \mathbf{W}^T\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{W}^T\boldsymbol{\lambda} - \mathbf{W}^T\mathbf{W}\mathbf{J}\mathbf{v}^{n+1} &= \mathbf{W}^T\mathbf{W}\mathbf{u}^\star \\ -\mathbf{W}^T\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G}\hat{\mathbf{p}} + \boldsymbol{\beta}^{-1}\mathbf{W}^T\boldsymbol{\lambda} - \mathbf{J}\mathbf{v}^{n+1} &= \mathbf{W}^T\mathbf{W}\mathbf{u}^\star \end{aligned} \tag{5.16}$$

Adding $\mathbf{G}^T$ times (5.16) to (5.13) yields

$$\mathbf{G}^T(\mathbf{I} - \mathbf{W}^T\mathbf{W})\boldsymbol{\beta}^{-1}\mathbf{G}\hat{\mathbf{p}} - \mathbf{G}^T\mathbf{J}\mathbf{v}^{n+1} = \mathbf{G}^T(\mathbf{I} - \mathbf{W}^T\mathbf{W})\mathbf{u}^\star.$$

Adding $\mathbf{J}^T\boldsymbol{\beta}$ times (5.16) to (5.15) yields

$$-\mathbf{J}^T\boldsymbol{\beta}\mathbf{W}^T\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G}\hat{\mathbf{p}} + (\Delta t\,\mathbf{D} - \mathbf{M} - \mathbf{J}^T\boldsymbol{\beta}\mathbf{J})\mathbf{v}^{n+1} = -\mathbf{M}\mathbf{v}^\star + \mathbf{J}^T\boldsymbol{\beta}\mathbf{W}^T\mathbf{W}\mathbf{u}^\star$$
$$-\mathbf{J}^T\mathbf{G}\hat{\mathbf{p}} + (\Delta t\,\mathbf{D} - \mathbf{M} - \mathbf{J}^T\boldsymbol{\beta}\mathbf{J})\mathbf{v}^{n+1} = -\mathbf{M}\mathbf{v}^\star + \mathbf{J}^T\boldsymbol{\beta}\mathbf{u}^\star$$

These together form a symmetric system similar to that of [125],

$$\begin{pmatrix} \mathbf{G}^T(\mathbf{I} - \mathbf{W}^T\mathbf{W})\boldsymbol{\beta}^{-1}\mathbf{G} & -\mathbf{G}^T\mathbf{J} \\ -\mathbf{J}^T\mathbf{G} & -\mathbf{J}^T\boldsymbol{\beta}\mathbf{J} - \mathbf{M} + \Delta t\,\mathbf{D} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{p}} \\ \mathbf{v}^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{G}^T(\mathbf{I} - \mathbf{W}^T\mathbf{W})\mathbf{u}^\star \\ -\mathbf{M}\mathbf{v}^\star + \mathbf{J}^T\boldsymbol{\beta}\mathbf{u}^\star \end{pmatrix}.$$

Both of these systems are symmetric indefinite. Although numerical methods exist for solving such systems, faster methods are available for systems which are symmetric positive definite. In this paper, we propose a novel approach which leads to a symmetric positive definite system.

We begin by forming the system

$$\begin{pmatrix} \mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{G} & -\mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{W}^T & \mathbf{0} \\ -\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G} & \mathbf{W}(\boldsymbol{\beta}^{-1} + \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T)\mathbf{W}^T & -\Delta t\,\mathbf{W}\mathbf{J}\mathbf{M}^{-1}\mathbf{D} \\ \mathbf{0} & \mathbf{M}^{-1}\mathbf{J}^T\mathbf{W}^T & \mathbf{I} - \Delta t\,\mathbf{M}^{-1}\mathbf{D} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{p}} \\ \boldsymbol{\lambda} \\ \mathbf{v}^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{G}^T\mathbf{u}^\star \\ \mathbf{W}\mathbf{J}\mathbf{v}^\star - \mathbf{W}\mathbf{u}^\star \\ \mathbf{v}^\star \end{pmatrix}, \tag{5.17}$$

where the equation, $-\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G}\hat{\mathbf{p}} + \mathbf{W}(\boldsymbol{\beta}^{-1} + \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T)\mathbf{W}^T\boldsymbol{\lambda} - \Delta t\,\mathbf{W}\mathbf{J}\mathbf{M}^{-1}\mathbf{D}\mathbf{v}^{n+1} = \mathbf{W}\mathbf{J}\mathbf{v}^\star - \mathbf{W}\mathbf{u}^\star$, is obtained by substituting (5.8) and (5.11) into (5.12). The first equation is (5.13), and the third equation is (5.15) multiplied through by $\mathbf{M}^{-1}$.

An obvious way to symmetrize this system is to multiply the third equation through by $-\Delta t\mathbf{D}$. Though this system is symmetric it is only positive semidefinite. The conditioning of the original structure system is typically quite good, since $\mathbf{I} - \Delta t\,\mathbf{M}^{-1}\mathbf{D}$ has eigenvalues clustered near one for small Δt. This symmetrized version looks more like $\mathbf{D}$, which will often have significantly worse conditioning.

A particularly interesting way of symmetrizing the system is to factor $\mathbf{D}$.

5.5.1 Factoring the Damping Matrix

If $\mathbf{D}$ is symmetric negative semidefinite, then it can be factored as $-\Delta t\,\mathbf{D} = \mathbf{C}^T\mathbf{C}$. We rely on this property of $\mathbf{D}$ to apply our method, noting that although there are instances where $\mathbf{D}$ will not have this property and in those cases our approach based on factoring $\mathbf{D}$ is not applicable, there are a wide range of models in which this property holds (see below). If $\mathbf{D}$ is a damping matrix, and the force is given by $\mathbf{f} = \mathbf{Ma} = -\nabla\phi + \mathbf{Dv}$, then the energy is $E = \phi + \frac{1}{2}\mathbf{v}^T\mathbf{Mv}$ and $\dot{E} = (\nabla\phi)^T\mathbf{v} + \mathbf{v}^T\mathbf{Ma} = (\nabla\phi)^T\mathbf{v} + \mathbf{v}^T(-\nabla\phi + \mathbf{Dv}) = \mathbf{v}^T\mathbf{Dv}$. Then, a non-positive change in energy leads to negative definiteness of $\mathbf{D}$. Furthermore, if $\mathbf{D}$ is decomposed into its symmetric and antisymmetric parts, $\frac{\mathbf{D}+\mathbf{D}^T}{2}$ and $\frac{\mathbf{D}-\mathbf{D}^T}{2}$ respectively, then $\mathbf{v}^T\left(\frac{\mathbf{D}-\mathbf{D}^T}{2}\right)\mathbf{v} = \mathbf{0}$ shows that the antisymmetric part does not contribute damping. Therefore, damping matrices gain little by containing asymmetric parts. These facts lead to the conclusion that for most purposes a damping matrix $\mathbf{D}$ can be considered to be symmetric negative semidefinite.

On the other hand, if you consider the backward Euler formulation at the beginning of Section 5.3, where $\mathbf{D}$ would be augmented by linearizations of elastic terms $\frac{\partial \mathbf{F}^n}{\partial \mathbf{x}}$ in the process of carrying out the Newton-Raphson iterations, then much of this would not be true. The current factorization is therefore limited to a semi-implicit treatment of the structure, where elastic terms are handled explicitly. Moreover, a fully implicit treatment would have the further complication of moving the locations of the constraints and changing $\mathbf{W}$ during the solve, which we do not address.

For the sake of exposition, consider the case of a linear spring which connects two particles $\mathbf{p}_1$ and $\mathbf{p}_2$. The magnitude of the damping force $\mathbf{f}$ is proportional to the difference in particle velocities along the spring direction $\mathbf{d}$, multiplied by a damping constant k_d. The force is then applied in opposite directions to $\mathbf{p}_1$ and $\mathbf{p}_2$. The damping matrix for this single spring would be

$$\mathbf{D} = -\Delta t k_d \begin{pmatrix} \mathbf{d} \\ -\mathbf{d} \end{pmatrix} \begin{pmatrix} \mathbf{d} \\ -\mathbf{d} \end{pmatrix}^T,$$

which can be factored into

$$\mathbf{C}_i = \sqrt{\Delta t k_d} \begin{pmatrix} \mathbf{d} \\ -\mathbf{d} \end{pmatrix}^T$$
$$\mathbf{D} = -\mathbf{C}_i^T \mathbf{C}_i.$$

For the case of n springs and an arbitrary number of particles, $\mathbf{C}$ is formed by stacking up rows of the form $\mathbf{C}_i$, and the same factorization applies.

Rayleigh damping is commonly used along with finite element constitutive models. In practice, we have found that our routines for applying damping forces are naturally partially factored. The implementation follows the framework of [81]. Below is a description of what the implementation looks like in the case of a single element and an isotropic constitutive model and how it lends itself to efficient factorization. In this framework, the deformation gradient is factored into its polar singular value decomposition as $\mathbf{F} = \mathbf{U}\hat{\mathbf{F}}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are rotations and $\hat{\mathbf{F}}$ is diagonal, in order to compute the elastic forces. Define two operators

$$G\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1 - \mathbf{x}_4 & \mathbf{x}_2 - \mathbf{x}_4 & \mathbf{x}_3 - \mathbf{x}_4 \end{pmatrix} \qquad S\begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 \end{pmatrix} = \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ -\mathbf{x}_1 - \mathbf{x}_2 - \mathbf{x}_3 \end{pmatrix}.$$

If $\mathbf{x}_m$ are the material space positions of the element, then $\mathbf{D}_m = G(\mathbf{x}_m)$ and $\mathbf{D}_s = G(\mathbf{x})$ are the locations of three vertices of the element relative to the fourth in material and world space. The damping force application routine consists of the steps

1. Compute the relative velocities, $\dot{\mathbf{D}}_s = G(\mathbf{v})$.

2. Compute the deformation gradient, $\dot{\mathbf{F}} = \dot{\mathbf{D}}_s \mathbf{D}_m^{-1}$.

3. Rotate the deformation gradient into the diagonalized coordinate system, $\hat{\dot{\mathbf{F}}} = \mathbf{U}^T \dot{\mathbf{F}} \mathbf{V}$.

4. Compute the first Piola-Kirchhoff stress in the diagonalized coordinate system, $\hat{\mathbf{P}} = \alpha \mathbf{tr}\left(\hat{\dot{\mathbf{F}}}\right)\mathbf{I} + \beta\hat{\dot{\mathbf{F}}} + \beta\hat{\dot{\mathbf{F}}}^T$, where α and β are parameters of the constitutive model.

5. Rotate back to compute the first Piola-Kirchhoff stress, $\mathbf{P} = \mathbf{U}\hat{\mathbf{P}}\mathbf{V}^T$.

6. Compute forces the element applies on three of the tetrahedron faces as $\mathbf{G} = -\mathbf{PN}$, where $\mathbf{N}$ are the area weighted face normals. These normals can be computed as $\mathbf{N} = -\frac{1}{2}\det(\mathbf{D}_m)\mathbf{D}_m^{-T}$. To see that this is correct, note that $\frac{1}{3}\mathbf{N}^T\mathbf{D}_m = -V\mathbf{I}$, where V is the volume of the tetrahedron. The negative sign is because $\mathbf{N}$ need to point outwards. Note that, if we computed it, $\mathbf{G}_4 = -\mathbf{PN}_4 = -\mathbf{P}(-\mathbf{N}_1 - \mathbf{N}_2 - \mathbf{N}_3) = -\mathbf{G}_1 - \mathbf{G}_2 - \mathbf{G}_3$.

7. Distribute the forces from faces to vertices. The force $\mathbf{f}_1$ at the first vertex is one third of the forces on the faces around it, or $\mathbf{f}_1 = \frac{1}{3}(\mathbf{G}_2 + \mathbf{G}_3 + \mathbf{G}_4) = -\frac{1}{3}\mathbf{G}_1$. Noting that the forces sum to one, the forces are finally obtained as $\mathbf{f} = S(-\frac{1}{3}\mathbf{G})$.

Several observations make the process of factoring this force application simpler. The first observation is that each of these steps can be regarded as a linear operator between vector spaces, where the matrices are taken to be flattened into vectors. In this way, the operators S and G are transposes, so that ignoring the factor of $-\frac{1}{3}$, the operator described by step 7 is the transpose of step 1. Similarly, step 6 is $\frac{1}{2}\det(\mathbf{D}_m)$ times the transpose of step 2. Step 3 is the transpose of step 5. Let $P(\alpha, \beta)$ represent the linear operator applied in step 4. The damping force application can be written as steps 1 through 3, followed by $-\frac{1}{6}\det(\mathbf{D}_m)P(\alpha, \beta)$, followed by the transpose of steps 1 through 3. Finally, the operation $\mathbf{C}$ is just steps 1 through 3, followed by the square root of $\frac{1}{6}\Delta t \det(\mathbf{D}_m)P(\alpha, \beta)$. It remains only to determine what this square root is.

Assume that the square root has the form $P(\alpha', \beta')$. Then,

$$
\begin{aligned}
P(\alpha', \beta')\hat{\dot{\mathbf{F}}} &= \alpha'\mathbf{tr}(\hat{\dot{\mathbf{F}}})\mathbf{I} + \beta'\hat{\dot{\mathbf{F}}} + \beta'\hat{\dot{\mathbf{F}}}^T \\
P(\alpha', \beta')^2\hat{\dot{\mathbf{F}}} &= \alpha'\mathbf{I}\mathbf{tr}(\alpha'\mathbf{tr}(\hat{\dot{\mathbf{F}}})\mathbf{I} + \beta'\hat{\dot{\mathbf{F}}} + \beta'\hat{\dot{\mathbf{F}}}^T) \\
&\qquad + \beta'(\alpha'\mathbf{tr}(\hat{\dot{\mathbf{F}}})\mathbf{I} + \beta'\hat{\dot{\mathbf{F}}} + \beta'\hat{\dot{\mathbf{F}}}^T) + \beta'(\alpha'\mathbf{tr}(\hat{\dot{\mathbf{F}}})\mathbf{I} + \beta'\hat{\dot{\mathbf{F}}} + \beta'\hat{\dot{\mathbf{F}}}^T)^T \\
&= (3\alpha'^2 + 4\alpha'\beta')\mathbf{tr}(\hat{\dot{\mathbf{F}}})\mathbf{I} + 2\beta'^2\hat{\dot{\mathbf{F}}} + 2\beta'^2\hat{\dot{\mathbf{F}}}^T \\
&= P(3\alpha'^2 + 4\alpha'\beta', 2\beta'^2)\hat{\dot{\mathbf{F}}} \\
\beta' &= \sqrt{\frac{1}{6}\Delta t \det(\mathbf{D}_m)}\left(\frac{1}{2}\sqrt{2\beta}\right) \\
\alpha' &= \sqrt{\frac{1}{6}\Delta t \det(\mathbf{D}_m)}\left(\frac{1}{3}\Big(\sqrt{3\alpha + 2\beta} - \sqrt{2\beta}\Big)\right)
\end{aligned}
$$

Since the operator $P(\alpha', \beta')$ is symmetric, this construction results in the operator $\mathbf{C}$, and routines to apply both $\mathbf{C}$ and $\mathbf{C}^T$ are readily constructed with only minor modifications to the existing code.

5.5.2 SPD Formulation

Assuming a factorization of the form $-\Delta t\,\mathbf{D} = \mathbf{C}^T\mathbf{C}$ allows us to left multiply the third row of (5.17) by $\mathbf{C}$ and subsequently factor $\mathbf{C}$ out of the third column into the solution vector to obtain

$$
\begin{pmatrix}
\mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{G} & -\mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{W}^T & \mathbf{0} \\
-\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G} & \mathbf{W}(\boldsymbol{\beta}^{-1} + \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T)\mathbf{W}^T & -\Delta t\,\mathbf{W}\mathbf{J}\mathbf{M}^{-1}\mathbf{D} \\
\mathbf{0} & \mathbf{M}^{-1}\mathbf{J}^T\mathbf{W}^T & \mathbf{I} - \Delta t\,\mathbf{M}^{-1}\mathbf{D}
\end{pmatrix}
\begin{pmatrix} \hat{\mathbf{p}} \\ \boldsymbol{\lambda} \\ \mathbf{v}^{n+1} \end{pmatrix}
=
\begin{pmatrix} \mathbf{G}^T\mathbf{u}^\star \\ \mathbf{W}\mathbf{J}\mathbf{v}^\star - \mathbf{W}\mathbf{u}^\star \\ \mathbf{v}^\star \end{pmatrix}
$$

$$
\begin{pmatrix}
\mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{G} & -\mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{W}^T & \mathbf{0} \\
-\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G} & \mathbf{W}(\boldsymbol{\beta}^{-1} + \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T)\mathbf{W}^T & \mathbf{W}\mathbf{J}\mathbf{M}^{-1}\mathbf{C}^T\mathbf{C} \\
\mathbf{0} & \mathbf{M}^{-1}\mathbf{J}^T\mathbf{W}^T & \mathbf{I} + \mathbf{M}^{-1}\mathbf{C}^T\mathbf{C}
\end{pmatrix}
\begin{pmatrix} \hat{\mathbf{p}} \\ \boldsymbol{\lambda} \\ \mathbf{v}^{n+1} \end{pmatrix}
=
\begin{pmatrix} \mathbf{G}^T\mathbf{u}^\star \\ \mathbf{W}\mathbf{J}\mathbf{v}^\star - \mathbf{W}\mathbf{u}^\star \\ \mathbf{v}^\star \end{pmatrix}
$$

$$
\begin{pmatrix}
\mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{G} & -\mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{W}^T & \mathbf{0} \\
-\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G} & \mathbf{W}(\boldsymbol{\beta}^{-1} + \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T)\mathbf{W}^T & \mathbf{W}\mathbf{J}\mathbf{M}^{-1}\mathbf{C}^T \\
\mathbf{0} & \mathbf{C}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{W}^T & \mathbf{I} + \mathbf{C}\mathbf{M}^{-1}\mathbf{C}^T
\end{pmatrix}
\begin{pmatrix} \hat{\mathbf{p}} \\ \boldsymbol{\lambda} \\ \hat{\mathbf{v}} \end{pmatrix}
=
\begin{pmatrix} \mathbf{G}^T\mathbf{u}^\star \\ \mathbf{W}\mathbf{J}\mathbf{v}^\star - \mathbf{W}\mathbf{u}^\star \\ \mathbf{C}\mathbf{v}^\star \end{pmatrix}
$$

where $\hat{\mathbf{v}} = \mathbf{C}\mathbf{v}^{n+1}$ represents the new unknowns. Note that the substitution of (5.11) into (5.12) to derive (5.17) allowed us to place a $\mathbf{D}$ into the third column of the coefficient matrix in (5.17), leading to the factorization of $\mathbf{C}$ out of the coefficient matrix and into the unknowns, and this was the reason for that extra substitution. The resulting system is obviously symmetric, and we can recover $\mathbf{v}^{n+1}$ from $\hat{\mathbf{v}}$ by replacing $\Delta t\mathbf{D}\mathbf{v}^{n+1}$ with $-\mathbf{C}^T\mathbf{C}\mathbf{v}^{n+1} = -\mathbf{C}^T\hat{\mathbf{v}}$ in (5.11)

$$\mathbf{M}\mathbf{v}^{n+1} = \mathbf{M}\mathbf{v}^{\star} - \mathbf{C}^T\hat{\mathbf{v}} - \mathbf{J}^T\mathbf{W}^T\boldsymbol{\lambda}.$$

The symmetrized system can be expressed as the sum of three matrices that are evidently symmetric positive semidefinite as

$$\begin{pmatrix} \mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{G} & -\mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{W}^T & \mathbf{0} \\ -\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G} & \mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{W}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} + \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{W}^T & \mathbf{W}\mathbf{J}\mathbf{M}^{-1}\mathbf{C}^T \\ \mathbf{0} & \mathbf{C}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{W}^T & \mathbf{C}\mathbf{M}^{-1}\mathbf{C}^T \end{pmatrix} + \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix},$$

from which the symmetric positive semidefiniteness of the coupled system follows.

To consider the rank of this matrix, note that any vector in the nullspace of the coupled system, consisting of three parts $\hat{\mathbf{p}}$, $\boldsymbol{\lambda}$, and $\hat{\mathbf{v}}$, must necessarily be in the nullspace of each of these three matrices. From the third matrix, we conclude that this requires $\hat{\mathbf{v}} = 0$. Assuming $\hat{\mathbf{v}} = 0$ and looking at the second matrix leads one to consider only requirements for $\boldsymbol{\lambda}$. The first and second matrices factor as

$$\begin{pmatrix} \mathbf{G}^T \\ -\mathbf{W} \\ \mathbf{0} \end{pmatrix} \boldsymbol{\beta}^{-1} \begin{pmatrix} \mathbf{G}^T \\ -\mathbf{W} \\ \mathbf{0} \end{pmatrix}^T + \begin{pmatrix} \mathbf{0} \\ \mathbf{W}\mathbf{J} \\ \mathbf{C} \end{pmatrix} \mathbf{M}^{-1} \begin{pmatrix} \mathbf{0} \\ \mathbf{W}\mathbf{J} \\ \mathbf{C} \end{pmatrix}^T.$$

For a vector to lie in the nullspace of the second matrix, it is necessary for $\mathbf{M}^{-1}\mathbf{J}^T\mathbf{W}^T\boldsymbol{\lambda} = 0$, which implies that the constraint impulses have no effect on any degrees of freedom of the structure. A nontrivial nullspace of the first matrix satisfies $\boldsymbol{\beta}^{-1}\mathbf{G}\hat{\mathbf{p}} - \boldsymbol{\beta}^{-1}\mathbf{W}^T\boldsymbol{\lambda} = 0$, which implies that the combination of constraints and pressures cancels to give no effect on the final fluid velocities $\mathbf{u}^{n+1}$ as can be seen by inspecting

(5.8). This immediately gives the cases $\boldsymbol{\beta}^{-1}\mathbf{G}\hat{\mathbf{p}} = 0$ (Neumann region) and $\mathbf{W}^T\boldsymbol{\lambda} = 0$ (redundant constraints), both of which will lead to nullspaces of the full coupled system by placing zeros in the remaining entries.

The right hand side of our equations must be in the range of the coefficient matrix in order for the system to be solvable. Comparing the form of our right hand side to the requirements that a nullspace must satisfy, we find that it is sufficient but not necessary condition for consistency for $\mathbf{v}^\star$ to contain no nullspace components of $\mathbf{M}^{-1}$ and $\mathbf{u}^\star$ to contain no nullspace components of $\boldsymbol{\beta}^{-1}$. A Neumann region will result in inconsistency if there is a net flux in $\mathbf{u}^\star$ through the region boundary, so we enforce consistency by summing up the net flux and subtracting it evenly from the $\mathbf{u}^\star$ on the region boundary faces. Clearly constraints which apply only to nullspace components of $\mathbf{M}^{-1}$ and $\boldsymbol{\beta}^{-1}$ will result in inconsistency and must be avoided. Fluid velocities which are constrained only to nullspace components of $\mathbf{M}^{-1}$ are in effect Neumann grid faces and must be taken into account in detection of Neumann regions. Isolated rigid bodies within a fluid region also do not prevent a region from being a Neumann region. A region that contains deformable bodies does not behave as a Neumann region, since the bodies can change volume to accommodate a net flux into the region.

5.5.3 Note On Forming SPD System

Many problems can be formulated as a KKT system of the form

$$\begin{pmatrix} \mathbf{M} + \mathbf{C}^T\mathbf{C} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix}, \tag{5.18}$$

where $\mathbf{M}$ is an SPD matrix that is easily inverted. Dissipative systems, such as fluid viscosity or structure damping, produce terms of the form $\mathbf{M} + \mathbf{C}^T\mathbf{C}$. Evolving such systems subject to constraints, such as the incompressibility constraint or the constraint that the solid and fluid velocities be equal at their interface, then leads to a KKT system in the form of (5.18). Multiplying the first row of (5.18) by $\mathbf{C}\mathbf{M}^{-1}$

produces

$$\mathbf{C}\mathbf{u} + \mathbf{C}\mathbf{M}^{-1}\mathbf{C}^T\mathbf{C}\mathbf{u} + \mathbf{C}\mathbf{M}^{-1}\mathbf{B}^T\boldsymbol{\lambda} = \mathbf{C}\mathbf{M}^{-1}\mathbf{b}. \tag{5.19}$$

Multiplying the first row of (5.18) by $\mathbf{B}\mathbf{M}^{-1}$ and simplifying using the second row produces

$$\mathbf{B}\mathbf{M}^{-1}\mathbf{C}^T\mathbf{C}\mathbf{u} + \mathbf{B}\mathbf{M}^{-1}\mathbf{B}^T\boldsymbol{\lambda} = \mathbf{B}\mathbf{M}^{-1}\mathbf{b}. \tag{5.20}$$

These are then combined into the SPD system using the definition $\hat{\mathbf{u}} = \mathbf{C}\mathbf{u}$ to produce

$$\begin{pmatrix} \mathbf{I} + \mathbf{C}\mathbf{M}^{-1}\mathbf{C}^T & \mathbf{C}\mathbf{M}^{-1}\mathbf{B}^T \\ \mathbf{B}\mathbf{M}^{-1}\mathbf{C}^T & \mathbf{B}\mathbf{M}^{-1}\mathbf{B}^T \end{pmatrix} \begin{pmatrix} \hat{\mathbf{u}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{C}\mathbf{M}^{-1}\mathbf{b} \\ \mathbf{B}\mathbf{M}^{-1}\mathbf{b} \end{pmatrix}. \tag{5.21}$$

Finally, $\mathbf{u}$ is obtained from the first row of (5.18) using

$$\mathbf{u} = \mathbf{M}^{-1}\mathbf{b} - \mathbf{M}^{-1}\mathbf{C}^T\hat{\mathbf{u}} - \mathbf{M}^{-1}\mathbf{B}^T\boldsymbol{\lambda}. \tag{5.22}$$

To demonstrate that this is in fact equivalent to (5.18), we show that we can derive (5.18) from $\hat{\mathbf{u}} = \mathbf{C}\mathbf{u}$, (5.21) and (5.22). The first row of (5.18) is obtained by substituting $\hat{\mathbf{u}} = \mathbf{C}\mathbf{u}$ into (5.22). The constraint row is obtained by subtracting the second row of (5.21) from $\mathbf{B}$ times (5.22). Thus, the indefinite KKT formulation is equivalent to the SPD formulation.

Note that the slightly more general form

$$\begin{pmatrix} \mathbf{M} + \mathbf{C}^T\mathbf{C} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{s} \end{pmatrix}, \tag{5.23}$$

can also be formed into an SPD system following the same formulation. This produces the system

$$\begin{pmatrix} \mathbf{I} + \mathbf{C}\mathbf{M}^{-1}\mathbf{C}^T & \mathbf{C}\mathbf{M}^{-1}\mathbf{B}^T \\ \mathbf{B}\mathbf{M}^{-1}\mathbf{C}^T & \mathbf{B}\mathbf{M}^{-1}\mathbf{B}^T \end{pmatrix} \begin{pmatrix} \hat{\mathbf{u}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{C}\mathbf{M}^{-1}\mathbf{b} \\ \mathbf{B}\mathbf{M}^{-1}\mathbf{b} - \mathbf{s} \end{pmatrix}. \tag{5.24}$$

Then, $\mathbf{u}$ can obtained as before.

5.6 Note On Conditioning

In the absence of coupling ($\mathbf{W} = 0$), the structure system looks like

$$\begin{aligned} \mathbf{v}^{n+1} &= \mathbf{v}^\star + \Delta t\, \mathbf{M}^{-1}\mathbf{D}\mathbf{v}^{n+1} \\ \mathbf{v}^{n+1} - \Delta t\, \mathbf{M}^{-1}\mathbf{D}\mathbf{v}^{n+1} &= \mathbf{v}^\star \\ (\mathbf{I} + \mathbf{M}^{-1}\mathbf{C}^T\mathbf{C})\mathbf{v}^{n+1} &= \mathbf{v}^\star \end{aligned}$$

The modified structure system looks like

$$(\mathbf{I} + \mathbf{C}\mathbf{M}^{-1}\mathbf{C}^T)\hat{\mathbf{v}} = \mathbf{C}\mathbf{v}^\star$$

These two systems have the same eigenvalues. To see this

$$\begin{aligned} (\mathbf{I} + \mathbf{M}^{-1}\mathbf{C}^T\mathbf{C})\mathbf{v} &= \alpha\mathbf{v} \\ \mathbf{C}(\mathbf{I} + \mathbf{M}^{-1}\mathbf{C}^T\mathbf{C})\mathbf{v} &= \mathbf{C}\alpha\mathbf{v} \\ (\mathbf{I} + \mathbf{C}\mathbf{M}^{-1}\mathbf{C}^T)\hat{\mathbf{v}} &= \alpha\hat{\mathbf{v}} \end{aligned}$$

so that if $\mathbf{v}$ is an eigenvector with eigenvalue α in the original system, then $\hat{\mathbf{v}}$ is an eigenvector of the modified system with the same eigenvalue. Note also that, though modifying the system in this way may introduce new eigenvalues (since the system may be larger), none of these eigenvalues will be visible to a Krylov solver. This is because the right hand side is of the form $\mathbf{C}\mathbf{v}^\star$ and application of the matrix results in vectors that are also of this form. The modified system may have fewer eigenvalues, however (caused by rank deficiency of $\mathbf{C}$). The important point is that this modification to the structure system will not adversely affect its convergence properties with a Krylov solver.

5.7 Viscosity

Applying viscosity as a separate step in the time evolution does not yield a fully coupled time integration scheme, since viscous forces cannot apply momentum to structures. This can be overcome by adding viscosity directly into the coupled formulation. With this in mind, we redefine the intermediate velocity

$$\mathbf{u}^\star = \mathbf{u}^n - \Delta t(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n + \frac{\Delta t}{\rho}\mathbf{f}. \tag{5.25}$$

Assume that the fluid viscosity μ is constant and ignore the coupling term. Then, the fluid momentum equation becomes

$$\rho\mathbf{u}^{n+1} = \rho\mathbf{u}^\star - \Delta t\nabla p + \Delta t\mu\nabla^2\mathbf{u}^{n+1}. \tag{5.26}$$

The viscous terms introduce a new gradient operator, which acts on face velocities. The gradient operator itself is discretized using central differences. The boundary conditions at structures are Dirichlet conditions, noting that these velocities are constrained implicitly using $\mathbf{W}$ and remain degrees of freedom rather than being moved to the right hand side as would normally be done. The discretization of $\mathbf{W}$ used above is not suitable for viscosity, however, since it does not couple to structures in the tangential direction. This is resolved by locating faces adjacent to those in $\mathbf{W}$ and adding them to $\mathbf{W}$ as shown in Figure 5.2. Domain boundary conditions can be discretized in the usual way. For simplicity of implementation, however, we treat domain walls as infinitely massive structures and add terms to $\mathbf{W}$, $\mathbf{J}$, and $\mathbf{M}^{-1}$ accordingly. This discretization is limited to first order accuracy since it uses the first order accurate constraint locations.

Let $\mathbf{G}_\mu = \sqrt{\Delta t\mu V}\nabla$ be the discretized gradient operator that acts on face velocities, where the scaling factors are pulled into the resulting matrix to simplify the presentation. Substituting this into (5.26) and adding back the coupling term produces the

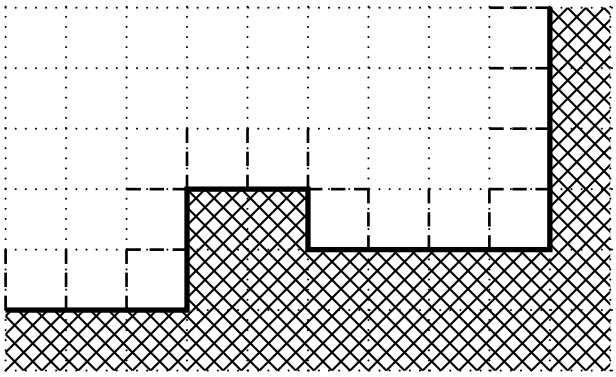


Figure 5.2: The hashed region is the rasterized structure, and the thick faces at its boundary are the original coupling faces in $\mathbf{W}$. For viscosity, the dashed faces are added to $\mathbf{W}$ to allow tangential forces to be exchanged between fluid and structure.

momentum equation

$$\boldsymbol{\beta}\mathbf{u}^{n+1} = \boldsymbol{\beta}\mathbf{u}^{\star} - \mathbf{G}\hat{\mathbf{p}} - \mathbf{G}_{\mu}^{T}\mathbf{G}_{\mu}\mathbf{u}^{n+1} + \mathbf{W}^{T}\boldsymbol{\lambda}. \tag{5.27}$$

Define $\hat{\mathbf{u}} = \mathbf{G}_{\mu}\mathbf{u}^{n+1}$. Then, multiplying (5.27) through by both $\mathbf{G}^{T}\boldsymbol{\beta}^{-1}$ (using (5.9)) and $\mathbf{G}_{\mu}\boldsymbol{\beta}^{-1}$ produces

$$\mathbf{G}^{T}\boldsymbol{\beta}^{-1}\mathbf{G}\hat{\mathbf{p}} + \mathbf{G}^{T}\boldsymbol{\beta}^{-1}\mathbf{G}_{\mu}^{T}\hat{\mathbf{u}} - \mathbf{G}^{T}\boldsymbol{\beta}^{-1}\mathbf{W}^{T}\boldsymbol{\lambda} = \mathbf{G}^{T}\mathbf{u}^{\star} \tag{5.28}$$

$$\mathbf{G}_{\mu}\boldsymbol{\beta}^{-1}\mathbf{G}\hat{\mathbf{p}} + (\mathbf{I} + \mathbf{G}_{\mu}\boldsymbol{\beta}^{-1}\mathbf{G}_{\mu}^{T})\hat{\mathbf{u}} - \mathbf{G}_{\mu}\boldsymbol{\beta}^{-1}\mathbf{W}^{T}\boldsymbol{\lambda} = \mathbf{G}_{\mu}\mathbf{u}^{\star}. \tag{5.29}$$

Multiplying (5.27) through by $\mathbf{W}\boldsymbol{\beta}^{-1}$ and substituting in (5.12) followed by (5.11), $-\Delta t\mathbf{D} = \mathbf{C}^{T}\mathbf{C}$ and $\mathbf{C}\mathbf{v}^{n+1} = \hat{\mathbf{v}}$ produces

$$\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G}\hat{\mathbf{p}} + \mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G}_{\mu}^{T}\hat{\mathbf{u}} - \mathbf{W}(\boldsymbol{\beta}^{-1} + \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{T})\mathbf{W}^{T}\boldsymbol{\lambda} - \mathbf{W}\mathbf{J}\mathbf{M}^{-1}\mathbf{C}^{T}\hat{\mathbf{v}} = \mathbf{W}\mathbf{u}^{\star} - \mathbf{W}\mathbf{J}\mathbf{v}^{\star}.$$

Combining these with (5.11) produces

$$\begin{pmatrix} \mathbf{G}^{T}\boldsymbol{\beta}^{-1}\mathbf{G} & \mathbf{G}^{T}\boldsymbol{\beta}^{-1}\mathbf{G}_{\mu}^{T} & -\mathbf{G}^{T}\boldsymbol{\beta}^{-1}\mathbf{W}^{T} & \mathbf{0} \\ \mathbf{G}_{\mu}\boldsymbol{\beta}^{-1}\mathbf{G} & \mathbf{I} + \mathbf{G}_{\mu}\boldsymbol{\beta}^{-1}\mathbf{G}_{\mu}^{T} & -\mathbf{G}_{\mu}\boldsymbol{\beta}^{-1}\mathbf{W}^{T} & \mathbf{0} \\ -\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G} & -\mathbf{W}\boldsymbol{\beta}^{-1}\mathbf{G}_{\mu}^{T} & \mathbf{W}(\boldsymbol{\beta}^{-1} + \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{T})\mathbf{W}^{T} & \mathbf{W}\mathbf{J}\mathbf{M}^{-1}\mathbf{C}^{T} \\ \mathbf{0} & \mathbf{0} & \mathbf{C}\mathbf{M}^{-1}\mathbf{J}^{T}\mathbf{W}^{T} & \mathbf{I} + \mathbf{C}\mathbf{M}^{-1}\mathbf{C}^{T} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{p}} \\ \hat{\mathbf{u}} \\ \boldsymbol{\lambda} \\ \hat{\mathbf{v}} \end{pmatrix} = \begin{pmatrix} \mathbf{G}^{T}\mathbf{u}^{\star} \\ \mathbf{G}_{\mu}\mathbf{u}^{\star} \\ \mathbf{W}\mathbf{J}\mathbf{v}^{\star} - \mathbf{W}\mathbf{u}^{\star} \\ \mathbf{C}\mathbf{v}^{\star} \end{pmatrix} \tag{5.30}$$

This equation has a structure very similar to what was obtained without viscosity, and it can be shown to be symmetric positive semidefinite in exactly the same way.

Stokes Flow When the Reynolds number is very small, the viscosity term dominates the inertial terms. Because viscosity and pressure are fully coupled in a single linear system, our scheme does not experience difficulties at high viscosity. In this case, one may also approximate the Navier-Stokes equations with those of Stokes flow by omitting the inertial terms. We can simulate the unsteady Stokes flow equations directly in our framework by omitting advection terms for the fluid (Step 6) and the structure (Step 4). This renders the entire first half of the update unnecessary, and our evolution scheme reduces to

1. $\mathbf{v}^\star = \mathbf{v}^n + \Delta t \mathbf{M}^{-1}\mathbf{F}_e(\mathbf{x}^{n+1})$
2. $\mathbf{u}^\star = \mathbf{u}^n + \frac{\Delta t}{\rho}\mathbf{f}$
3. Coupled solve with known quantities $\mathbf{u}^\star$, $\mathbf{v}^\star$, and $\mathbf{x}^{n+1}$ and unknown quantities $\mathbf{u}^{n+1}$, $\mathbf{p}$, and $\mathbf{v}^{n+1}$

We demonstrate our ability to simulate with high viscosity and Stokes flow in Section 5.11.2.

As discussed in [125], the indefinite system of equations derived for fluid-structure interaction has many similarities to the Stokes equations as both are dissipative systems combined with constraints (see Section 5.5.3). Indeed, we produce an SPD formulation for both. If no structures are present then $\mathbf{W} = \mathbf{0}$, $\mathbf{C} = \mathbf{0}$ and $\mathbf{J} = \mathbf{0}$, and (5.30) simplifies to

$$\begin{pmatrix} \mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{G} & \mathbf{G}^T\boldsymbol{\beta}^{-1}\mathbf{G}_\mu^T \\ \mathbf{G}_\mu\boldsymbol{\beta}^{-1}\mathbf{G} & \mathbf{I} + \mathbf{G}_\mu\boldsymbol{\beta}^{-1}\mathbf{G}_\mu^T \end{pmatrix} \begin{pmatrix} \hat{\mathbf{p}} \\ \hat{\mathbf{u}} \end{pmatrix} = \begin{pmatrix} \mathbf{G}^T\mathbf{u}^\star \\ \mathbf{G}_\mu\mathbf{u}^\star \end{pmatrix}. \tag{5.31}$$

This system provides an SPD treatment of unsteady Stokes flow. If convection is handled explicitly, this formulation also provides an SPD treatment of Navier-Stokes.

Fully implicit Navier Stokes We could take a fully implicit approach to the full nonlinear Navier-Stokes equations by using a Newton-Raphson iteration method, where we linearize at each Newton step to get a system of the form in (5.31). The equation to be iterated is

$$\mathbf{u}_{(i+1)}^{n+1} = \mathbf{u}^n - \Delta t(\mathbf{u}_{(i)}^{n+1} \cdot \nabla)\mathbf{u}_{(i)}^{n+1} + \frac{\Delta t}{\rho}(-\nabla \mathbf{p} + \mu \nabla^2 \mathbf{u}_{(i+1)}^{n+1} + \mathbf{f}),$$

where $\mathbf{u}_{(i)}^{n+1}$ is the time t^{n+1} fluid velocity after the i^{th} Newton iteration and we let $\mathbf{u}_{(0)}^{n+1} = \mathbf{u}^n$. The inner iterations are performed by solving the symmetric and positive definite system (5.31). For each outer iteration, a new system is set up using the new estimate of the final velocity to recompute the advection term. A similar approach could be explored for treatment of fully implicit coupled fluid-structure evolution.

5.8 Note on Stability

We have found the proposed monolithic implicit coupling scheme described in this paper to be quite stable. We have never observed stability problems due to this simple choice. In the example from Figure 5.8, we consider the structure discretization to be exact and refine only the fluid discretization. This results in a grid spacing much smaller than the lengths of edges in the structure mesh, which has never been observed to cause stability problems. We also tried running examples with a fixed resolution fluid mesh, refining the structure so that the grid spacing was over 30 times the lengths of the structure edges, again without noticing any stability problems. We have tried making structures a billion times less dense than the fluid and a billion times more dense than the fluid, and neither modification lead to stability problems.

We also note that solving the coupled linear system and applying the resulting update to the velocities of the structure and the fluid has the property that it never increases total kinetic energy. Since positions are not changed during this process, potential

energy is unchanged. Let

$$\mathbf{K} = \begin{pmatrix} \mathbf{G}^T & \mathbf{0} \\ \mathbf{G}_\mu & \mathbf{0} \\ -\mathbf{W} & \mathbf{WJ} \\ \mathbf{0} & \mathbf{C} \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} \boldsymbol{\beta} & \mathbf{0} \\ \mathbf{0} & \mathbf{M} \end{pmatrix} \qquad \mathbf{P} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix}$$

$$\mathbf{w} = \begin{pmatrix} \mathbf{u}^\star \\ \mathbf{v}^\star \end{pmatrix} \qquad \mathbf{w}^{n+1} = \begin{pmatrix} \mathbf{u}^{n+1} \\ \mathbf{v}^{n+1} \end{pmatrix} \qquad \mathbf{z} = \begin{pmatrix} \hat{\mathbf{p}} \\ \hat{\mathbf{u}} \\ \boldsymbol{\lambda} \\ \hat{\mathbf{v}} \end{pmatrix}$$

(5.30) reduces to $(\mathbf{KB}^{-1}\mathbf{K}^T + \mathbf{P})\mathbf{z} = \mathbf{Kw}$. The new velocities are $\mathbf{w}^{n+1} = \mathbf{w} - \mathbf{B}^{-1}\mathbf{K}^T\mathbf{z}$. Finally, the change in kinetic energy is

$$\begin{aligned} 2KE^{n+1} - 2KE^\star &= (\mathbf{w} - \mathbf{B}^{-1}\mathbf{K}^T\mathbf{z})^T\mathbf{B}(\mathbf{w} - \mathbf{B}^{-1}\mathbf{K}^T\mathbf{z}) - \mathbf{w}^T\mathbf{Bw} \\ &= -2\mathbf{w}^T\mathbf{K}^T\mathbf{z} + (\mathbf{K}^T\mathbf{z})^T\mathbf{B}^{-1}(\mathbf{K}^T\mathbf{z}) \\ &= -2\mathbf{z}^T(\mathbf{KB}^{-1}\mathbf{K}^T + \mathbf{P})\mathbf{z} + \mathbf{z}^T\mathbf{KB}^{-1}\mathbf{K}^T\mathbf{z} \\ &= -\mathbf{z}^T(\mathbf{KB}^{-1}\mathbf{K}^T + 2\mathbf{P})\mathbf{z} \leq 0 \end{aligned}$$

This does not imply that we always decrease kinetic energy over a full time step, since elastic forces added to the structures or gravity added to fluids may both increase kinetic energy. In particular, we are still explicit on elastic forces, and this continues to impose a CFL restriction. On the other hand, if there is no potential energy (e.g., rigid bodies without forces between them), and the remaining steps in the time integration scheme make a similar guarantee on kinetic energy, the resulting scheme would be unconditionally stable. We also note that if there is neither damping nor viscosity, then $\mathbf{P} = \mathbf{0}$, and the process of obtaining the final velocities $\mathbf{w}^{n+1}$ from the intermediate velocities $\mathbf{w}$ is described by the mass-symmetric projection matrix $\mathbf{I} - \mathbf{B}^{-1}\mathbf{K}^T(\mathbf{KB}^{-1}\mathbf{K}^T)^{-1}\mathbf{K}$, which is exactly the property that would be desired in this case.

5.9 Minimization Formulation

It was pointed out in [16] that the pressure projection could be formulated as a kinetic energy minimization problem. By allowing fluid pressure to exert forces on rigid bodies in contact with the fluid, the pressures in addition to enforcing incompressibility in the fluid also apply coupling exchanges between fluid and solid. The total kinetic energy includes the kinetic energy of the fluid and structures. As in the case with no solids, the pressures are computed by choosing pressures to minimize the kinetic energy. This raises the question of whether the variational analogy also extends to the FSI treatment presented here. There is in fact a quantity being minimized, but that quantity is not quite the kinetic energy.

The final velocities are obtained using the coupled system and application rule

$$(\mathbf{K}\mathbf{B}^{-1}\mathbf{K}^T + \mathbf{P})\mathbf{z} = \mathbf{K}\mathbf{w} \qquad \mathbf{w}^{n+1} = \mathbf{w} - \mathbf{B}^{-1}\mathbf{K}^T\mathbf{z}. \tag{5.32}$$

The total kinetic energy of the system is

$$KE = \frac{1}{2}(\mathbf{w}^{n+1})^T\mathbf{B}\mathbf{w}^{n+1} = \frac{1}{2}(\mathbf{w} - \mathbf{B}^{-1}\mathbf{K}^T\mathbf{z})^T\mathbf{B}(\mathbf{w} - \mathbf{B}^{-1}\mathbf{K}^T\mathbf{z}). \tag{5.33}$$

In [16], the quantities being minimized over were the pressures. That is, the forces were chosen to minimize kinetic energy. In the case of our formulation, the forces are $\mathbf{z}$. Recall that $\mathbf{z}$ contains $\hat{\mathbf{p}}$, the pressures, $\hat{\mathbf{u}}$, the viscosity forces, $\boldsymbol{\lambda}$, the coupling forces which exchange forces between fluid and structures as pressures did in [16], and $\hat{\mathbf{v}}$, which describes the structure's constitutive forces. Differentiating KE with respect to $\mathbf{z}$ produces

$$\mathbf{0} = \frac{\partial KE}{\partial \mathbf{z}} = -(\mathbf{B}^{-1}\mathbf{K}^T)^T\mathbf{B}(\mathbf{w} - \mathbf{B}^{-1}\mathbf{K}^T\mathbf{z}) = -\mathbf{K}\mathbf{w} + \mathbf{K}\mathbf{B}^{-1}\mathbf{K}^T\mathbf{z}. \tag{5.34}$$

This leads to the system

$$\mathbf{K}\mathbf{B}^{-1}\mathbf{K}^T\mathbf{z} = \mathbf{K}\mathbf{w}, \tag{5.35}$$

which is not quite the correct coupling system.

This differs from the correct system by the term $\mathbf{Pz}$. This suggests that one should instead minimize the modified energy

$$E = KE + \frac{1}{2}\mathbf{z}^T\mathbf{Pz} = \frac{1}{2}(\mathbf{w}^{n+1})^T\mathbf{B}\mathbf{w}^{n+1} + \frac{1}{2}\mathbf{z}^T\mathbf{Pz}. \tag{5.36}$$

Observe that, as with KE, the quantity E is strictly nonnegative and quadratic in $\mathbf{z}$, making the minimization well-posed. Differentiating produces

$$\mathbf{0} = \frac{\partial E}{\partial \mathbf{z}} = \frac{\partial KE}{\partial \mathbf{z}} + \mathbf{Pz} = -\mathbf{Kw} + \mathbf{K}\mathbf{B}^{-1}\mathbf{K}^T\mathbf{z} + \mathbf{Pz}. \tag{5.37}$$

This leads to the correct coupling system

$$(\mathbf{K}\mathbf{B}^{-1}\mathbf{K}^T + \mathbf{P})\mathbf{z} = \mathbf{Kw}. \tag{5.38}$$

That the kinetic energy should be part of the minimization is not surprising, as this is part of the classical Lagrangian. The interpretation of the $\frac{1}{2}\mathbf{z}^T\mathbf{Pz}$ term is much less clear. Since the $\mathbf{P}$ matrix is associated with damping terms (the vector $\mathbf{z}$ contains constraint forces $\hat{\mathbf{p}}$ and $\boldsymbol{\lambda}$ and damping forces $\hat{\mathbf{v}}$ and $\hat{\mathbf{u}}$, and $\mathbf{P}$ removes the contributions from the constraint forces), the additional term would appear to be some sort of damping energy. Indeed

$$\frac{1}{2}\mathbf{z}^T\mathbf{Pz} = \frac{1}{2}\hat{\mathbf{u}}^T\hat{\mathbf{u}} + \frac{1}{2}\hat{\mathbf{v}}^T\hat{\mathbf{v}} \tag{5.39}$$

Recall that $\hat{\mathbf{v}} = \mathbf{C}\mathbf{v}^{n+1}$ and $\hat{\mathbf{u}} = \mathbf{G}_\mu\mathbf{u}^{n+1}$. Substituting this in produces

$$\frac{1}{2}\mathbf{z}^T\mathbf{Pz} = \frac{1}{2}(\mathbf{v}^{n+1})^T\mathbf{C}^T\mathbf{C}\mathbf{v}^{n+1} + \frac{1}{2}(\mathbf{u}^{n+1})^T\mathbf{G}_\mu^T\mathbf{G}_\mu\mathbf{u}^{n+1}. \tag{5.40}$$

Next, recall $\mathbf{C}^T\mathbf{C} = -\Delta t\mathbf{D}$, so that $\mathbf{C}^T\mathbf{C}\mathbf{v}^{n+1} = -\Delta t\mathbf{f}_{damp}$ and similarly $\mathbf{G}_\mu^T\mathbf{G}_\mu\mathbf{u}^{n+1} = -\Delta t\mathbf{f}_{visc}$, where $\mathbf{f}_{damp}$ and $\mathbf{f}_{visc}$ represent the damping and viscosity forces at time t^{n+1}.

The extra term is now

$$\frac{1}{2}\mathbf{z}^T\mathbf{P}\mathbf{z} = -\frac{1}{2}\Delta t(\mathbf{v}^{n+1})^T\mathbf{f}_{damp} - \frac{1}{2}\Delta t(\mathbf{u}^{n+1})^T\mathbf{f}_{visc}. \tag{5.41}$$

Then, $\Delta t\mathbf{v}^{n+1}$ is the change in position, so that $W_{damp} = \Delta t(\mathbf{v}^{n+1})^T\mathbf{f}_{damp}$ is the work done by structure damping and similarly $W_{visc} = \Delta t(\mathbf{u}^{n+1})^T\mathbf{f}_{visc}$ is the work done by fluid viscosity. The quantity being minimized is finally

$$E = KE - \frac{1}{2}(W_{damp} + W_{visc}). \tag{5.42}$$

This is consistent with [16], since they did not include fluid viscosity or structure constitutive forces so that in their case $W_{damp} = W_{visc} = 0$.

5.10 Structure with Constraints

Consider a backward Euler velocity update with constraints, such as incompressibility (Section 3.6), contact (Section 3.4 and Section 4.7.3), and poststabilization (Section 4.7.3).

$$\begin{aligned} \mathbf{M}\mathbf{v}^{n+1} &= \mathbf{M}\mathbf{v}^\star + \Delta t\,\mathbf{D}\mathbf{v}^{n+1} - \mathbf{H}^T\boldsymbol{\lambda} \\ \mathbf{H}\mathbf{v}^{n+1} &= \mathbf{s} \end{aligned} \tag{5.43}$$

Here, $\mathbf{H}\mathbf{v}^{n+1} = \mathbf{s}$ contains all of the constraints. Comparing with (5.24) results in the system

$$\begin{pmatrix} \mathbf{I} + \mathbf{C}\mathbf{M}^{-1}\mathbf{C}^T & \mathbf{C}\mathbf{M}^{-1}\mathbf{H}^T \\ \mathbf{H}\mathbf{M}^{-1}\mathbf{C}^T & \mathbf{H}\mathbf{M}^{-1}\mathbf{H}^T \end{pmatrix} \begin{pmatrix} \hat{\mathbf{u}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{C}\mathbf{v}^\star \\ \mathbf{H}\mathbf{v}^\star - \mathbf{s} \end{pmatrix},$$

where $\mathbf{v}^{n+1}$ is obtained by substituting into (5.43).

Note that the presence of inhomogeneous constraints $\mathbf{s} \neq \mathbf{0}$ was not considered in the original formulation, though it could be included the same way as here. With this extra term, the system will be inconsistent only if $\mathbf{s}$ is not in the column space of

H. This can be demonstrated using the same logic as before. The meaning of this inconsistency is clear, since in this case the constraint $\mathbf{H}\mathbf{v}^{n+1} = \mathbf{s}$ cannot possibly be satisfied. On the other hand, if the constraint can be satisfied, then the resulting consistency of the linear system implies that the constraint will be satisfied.

5.11 Examples

The SPD formulation merely gives an analytically equivalent expression of the indefinite form of the equations. What we expect to illustrate in these examples is that the method is numerically stable. We examine several common examples from the fluid structure interaction literature to demonstrate the convergence properties of our method.

We choose our time step size to be the minimum of the time step size required for fluids alone and the time step size required for structures alone. The fluid time step size is computed using a CFL number of 0.9 for all of our examples. Except in the stability test described below, the structures in our examples are not refined, so the time step size computed for the structures will not change significantly from resolution to resolution. For higher resolutions, the fluid time step restriction will dominate. We note that since all of our examples converge to nonzero velocities under refinement, we effectively refine with $\Delta t \propto \Delta x$. Thus, the first order convergence demonstrated in our examples implies that our scheme is at least first order accurate in space and time. We do not test convergence order separately for space and time, since we do not expect to achieve better than first order.

5.11.1 Analytic example

Figure 5.3 shows a simple setup designed to mimic an infinite channel with a rigid structure flowing through it and centered in the channel. The density of the fluid is $\rho = 100kg\,m^{-2}$, and the mass of the rigid body is $M = 150kg$. The height of

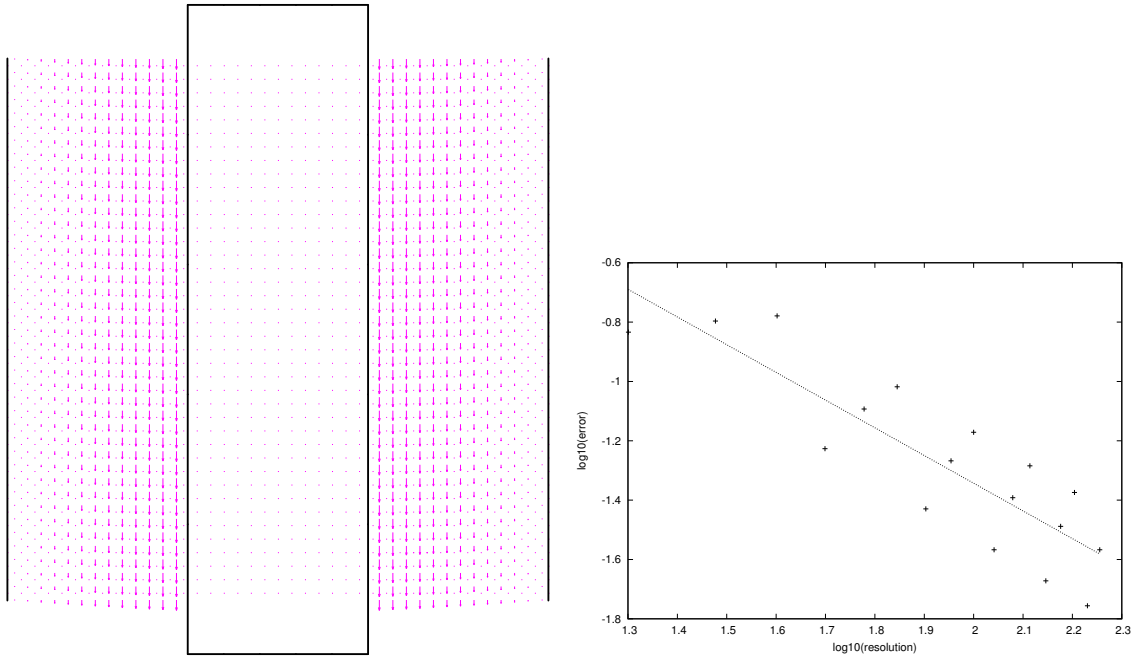
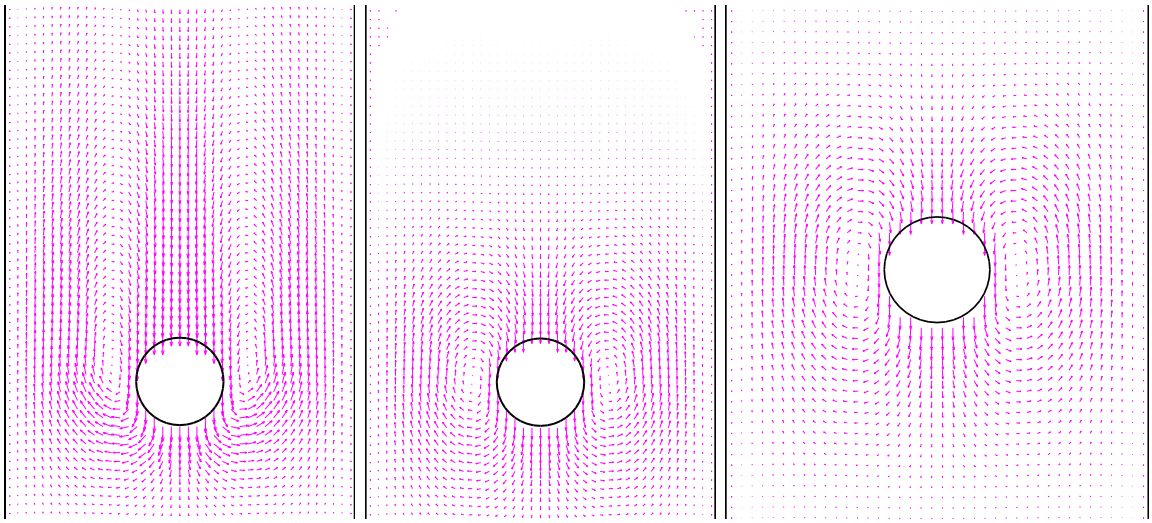


Figure 5.3: A structure flows through a channel in an analytic test. The errors are computed as the difference between the steady state velocity of the moving structure and the analytic solution. The error depends on the grid resolution mod 3 since this value determines how the structure aligns with the domain. The convergence order of the errors is fit as .93.

the fluid domain is $h = 1m$, and the total domain width of $1m$ is divided into two fluid regions with width $w = \frac{1}{3}m$ with the central $\frac{1}{3}m$ occupied by the structure. The fluid viscosity is $\mu = 100kg\,s^{-1}$. Both the fluid and structure experience gravity $g = -9.8ms^{-1}$. We elongate the rigid body to extend slightly beyond the domain and periodically reset its position, providing the illusion of an infinitely long rigid body and allowing the simulation to be run long enough to reach steady state. Altering the dimensions of the structure does not affect the analytic solution, provided its mass is not altered.

We derive the analytic solution by considering the structure to be a boundary condition to the fluid and then solving for the structure's velocity using conservation of energy. The analytic velocity profile for the left fluid region $\Omega = [0, w] \times [0, h]$



(a) N-S, $\mu = 0.1\,kg\,s^{-1}$ (b) N-S, $\mu = 0.5\,kg\,s^{-1}$ (c) Stokes flow, $\mu = 0.1\,kg\,s^{-1}$

Figure 5.4: A steady state velocity field is computed using the Navier-Stokes equations (N-S) or the equations for Stokes flow.

is $u = 0$ and $v = \frac{\rho g}{2\mu}x(x - w) + \frac{v_s}{w}x$, where the v is the vertical fluid velocity component and v_s is the vertical component of the structure velocity. The fluid experiences no pressure gradient. The only unknown remaining is v_s, the velocity of the structure. Taking into account the symmetry of the system, conservation of energy implies that the viscous dissipation equals the total work applied by gravity, so that $2\int_\Omega (\tau \cdot \nabla)\mathbf{u}\, dV = 2\int_\Omega \rho g v\, dV + Mgv_s$. Solving this equation yields the analytic solution $v_s = -(M + \rho w h)\frac{gw}{2h\mu}$.

5.11.2 Rigid cylinder falling under gravity

Motivated by [165, 166], a rigid cylinder is allowed to fall under an externally applied gravitational force. The fluid domain has dimensions $2L \times 8L$, and the cylinder has radius r. If the fluid density is ρ_f and the solid density is ρ_s, then the terminal velocity is [165]

$$v_{term} = \frac{(\rho_s - \rho_f)gr^2}{4\mu}\left(-\ln\left(\frac{r}{L}\right) - 0.9157 + 1.7244\left(\frac{r}{L}\right)^2 - 1.7302\left(\frac{r}{L}\right)^4\right).$$

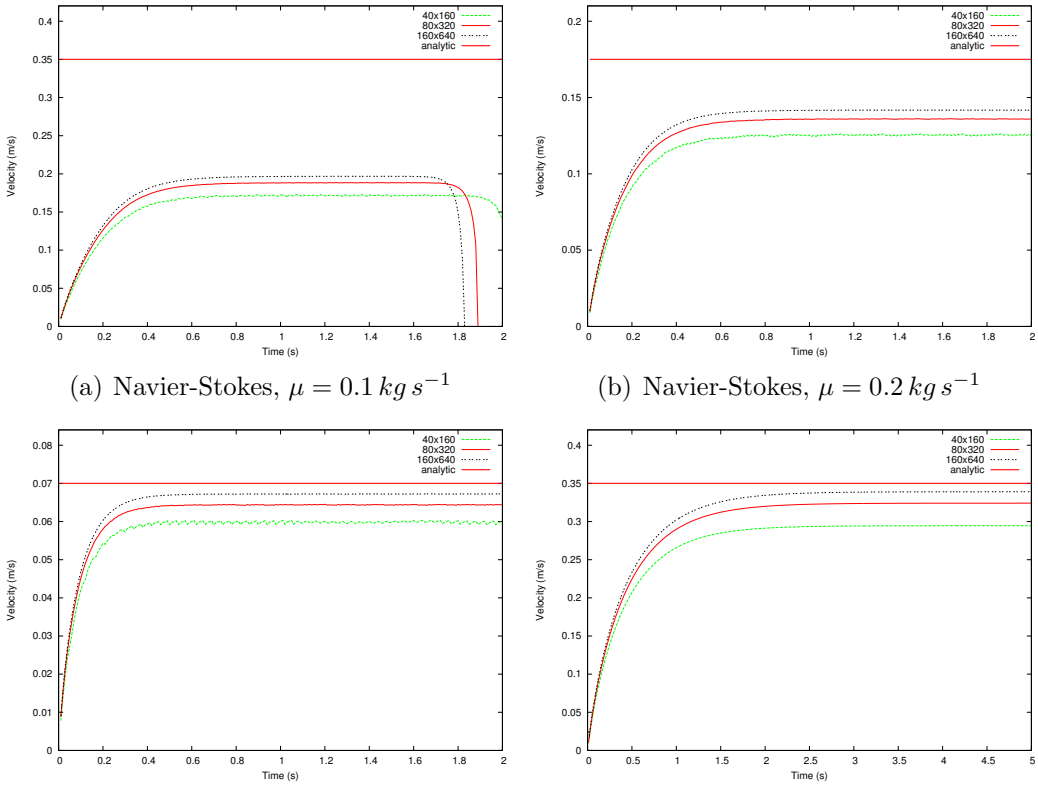
It is instructive to note the origin of this analytic solution. Consider a fluid domain consisting of an infinite 2D channel with parallel walls separated by a distance of $2L$. In the middle of this channel is a cylinder with radius a moving with a constant velocity v_s along the axis of the channel. Under the simplifying assumption of Stokes flow, the resistive force experienced by the cylinder is approximately [67]

$$F = \frac{4\pi\mu v_s}{-\ln\left(\frac{r}{L}\right) - 0.9157 + 1.7244\left(\frac{r}{L}\right)^2 - 1.7302\left(\frac{r}{L}\right)^4}.$$

Note that because of the Stokes flow assumption, fluid advection and movement of the cylinder are assumed to be insignificant and dropped. The steady state solid velocity is obtained by balancing the resistive force against the net gravitational force $F = (\rho_s - \rho_f)g\pi r^2$ on the solid (taking into account buoyancy).

We use the same parameters as [165]: $g = 9.8\,ms^{-1}$, $r = 5 \cdot 10^{-3}\,m$, $L = 2 \cdot 10^{-2}\,m$, $\mu = 0.1\,kg\,s^{-1}$, $\rho_f = 1 \cdot 10^3\,kg\,m^{-2}$, and $\rho_s = 2 \cdot 10^3\,kg\,m^{-2}$. The top of the fluid domain has a zero pressure Dirichlet boundary condition, and the sides and bottom are no-slip walls. Our results are shown in Figure 5.5(a). We note that our results do not agree with those in [165], and we note that their velocity profile does not appear to reach a terminal velocity. We converge to a value significantly lower than the analytic solution. The analytic solution was derived under the assumption of Stokes flow, and the given parameters, $Re = 35$, which is well outside the Stokes flow regime. We also ran with higher viscosities: $\mu = 0.2\,kg\,s^{-1}$ (Figure 5.5(b)), $\mu = 0.5\,kg\,s^{-1}$ (Figure 5.5(c)), $\mu = 1.0\,kg\,s^{-1}$, $\mu = 2.0\,kg\,s^{-1}$, $\mu = 5.0\,kg\,s^{-1}$, and $\mu = 10.0\,kg\,s^{-1}$. We obtain close agreement with the analytic solution for $\mu = 0.5\,kg\,s^{-1}$ and higher viscosities. We also repeated the the original test $\mu = 0.1\,kg\,s^{-1}$ using Stokes flow and converge to the analytic terminal velocity as shown in Figure 5.5(d). The results from all of these simulations are summarized in Table 5.1, and the steady state velocities are shown in Figure 5.4.

The velocity profile shown in Figure 7 of [166] does show a faster initial increase in velocity followed by the velocity profile reaching a plateau, which is consistent with the behavior we observe. Their method, like ours, is first order accurate, and their



(a) Navier-Stokes, $\mu = 0.1\, kg\, s^{-1}$

(b) Navier-Stokes, $\mu = 0.2\, kg\, s^{-1}$

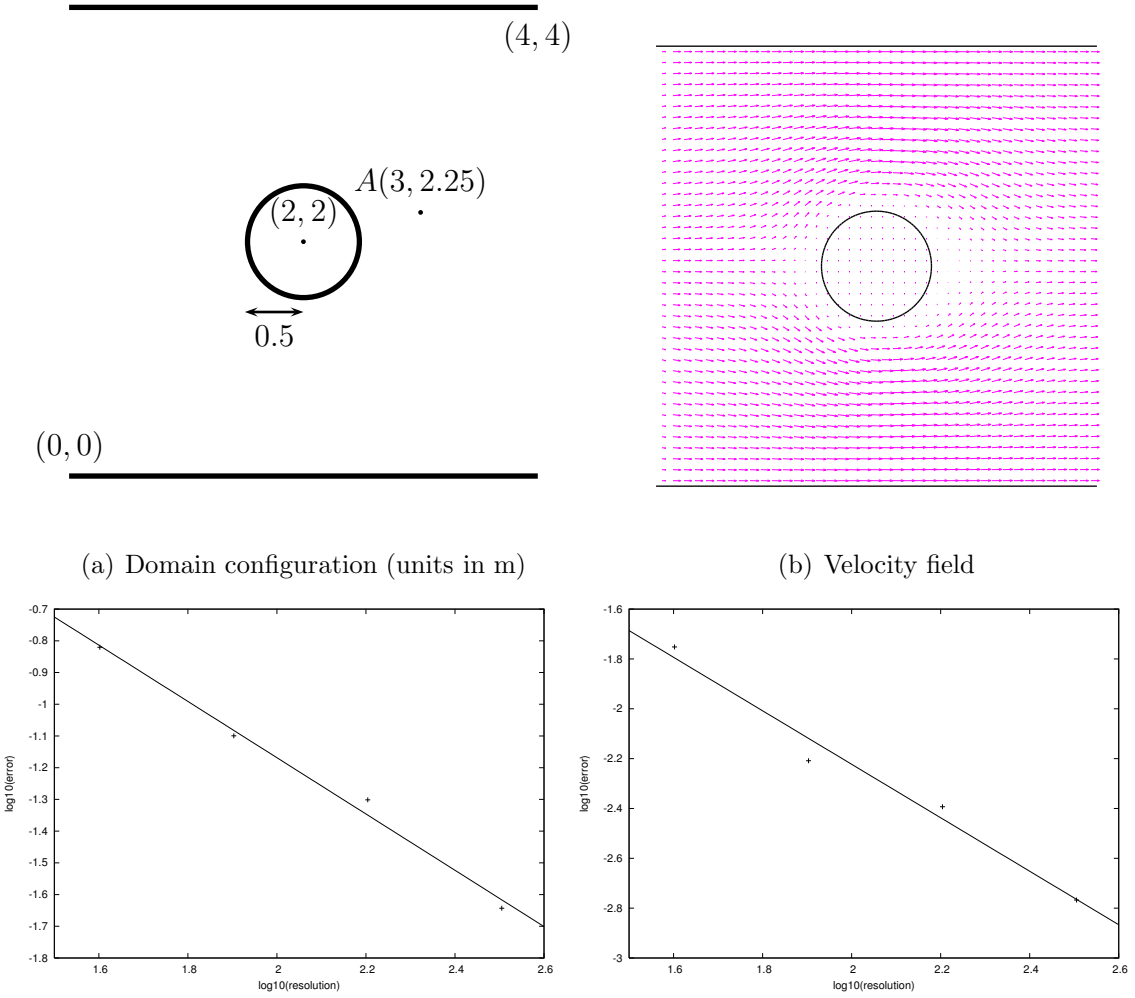
(c) Navier-Stokes, $\mu = 0.5\, kg\, s^{-1}$

(d) Stokes flow, $\mu = 0.1\, kg\, s^{-1}$

Figure 5.5: Velocity of a falling cylinder in a channel over time.

Eqn.	μ ($kg\, s^{-1}$)	Terminal Velocities (ms^{-1}) Analytic	40×160	80×320	160×640	Convergence Order	
N-S	0.1	-0.3501	-0.1716	-0.1882	-0.1966	-	-
N-S	0.2	-0.1750	-0.1254	-0.1359	-0.1417	-	-
N-S	0.5	-0.07002	-0.05983	-0.06438	-0.06721	0.85	1.01
N-S	1.0	-0.03501	-0.03052	-0.03264	-0.03399	0.92	1.22
N-S	2.0	-0.01750	-0.01533	-0.01635	-0.01702	0.91	1.26
N-S	5.0	-0.007002	-0.006148	-0.006547	-0.006828	0.91	1.38
N-S	10.0	-0.003501	-0.003013	-0.003279	-0.003417	1.14	1.40
Stokes	0.1	-0.3501	-0.2945	-0.3241	-0.3390	1.10	1.23

Table 5.1: Terminal velocity of a falling cylinder and convergence orders.



(a) Domain configuration (units in m)

(b) Velocity field

(c) Convergence of $\mathbf{u}_1$ at A.

(d) Convergence of $\mathbf{u}_2$ at A.

Figure 5.6: Laminar flow around a fixed cylinder.

figure shows two velocity profiles for each of two methods corresponding to a factor-of-four refinement. This suggests that the close fit shown in their highest resolution may be coincidental and that their scheme will actually converge to a value somewhat higher than the predicted value.

5.11.3 Laminar flow past a cylinder

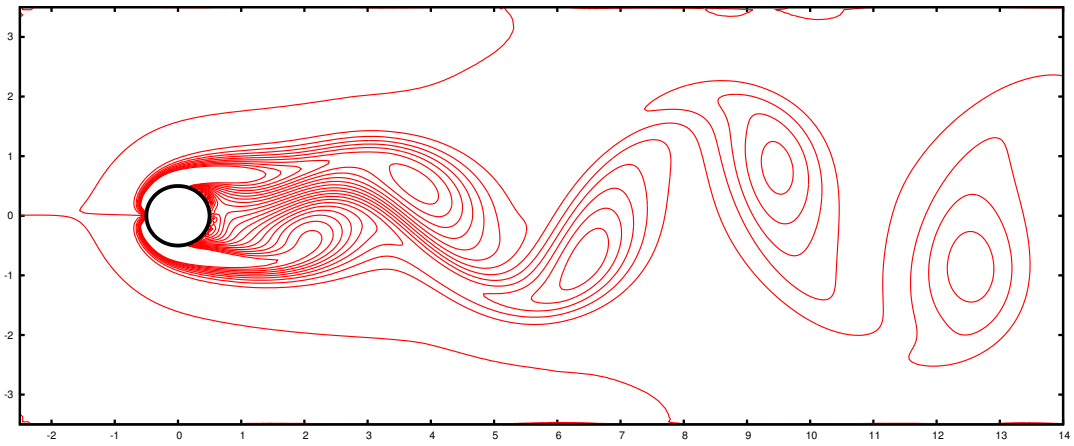
In this example, we consider a simple one-way coupled setup and directly compare our technique against a different scheme. For the comparison scheme, we use standard Chorin splitting [37]. We then choose a sample point at a time and position which exhibits dynamic behavior and demonstrate that the two schemes converge to the same velocity at that point. This demonstrates that we are consistent with an existing scheme from the literature in the simpler case where structures are only one-way coupled to the fluid.

For this comparison we use laminar flow across a fixed cylinder. The domain is $[0\,m, 4\,m] \times [0\,m, 4\,m]$ with a cylinder of radius $0.5\,m$ located at $(2, 2)$. The cylinder has no-slip boundary conditions. The top and bottom walls have slip boundary conditions. The right wall has the condition $p = 0$, and the left wall has the boundary condition $\mathbf{u}_1 = 1$ and a slipping boundary condition for $\mathbf{u}_2$. The density is $\rho = 1kg\,m^{-2}$, and the viscosity is $\mu = 0.1\,kg\,s^{-1}$. The initial fluid velocity is $\mathbf{u} = \mathbf{0}$. The setup is illustrated in Figure 5.6(a). The fluid velocity is sampled at the point $(3, 2.25)$ at time $0.5s$ using linear interpolation. The fluid velocity is far from steady state at this time, so this comparison also tests dynamic behavior.

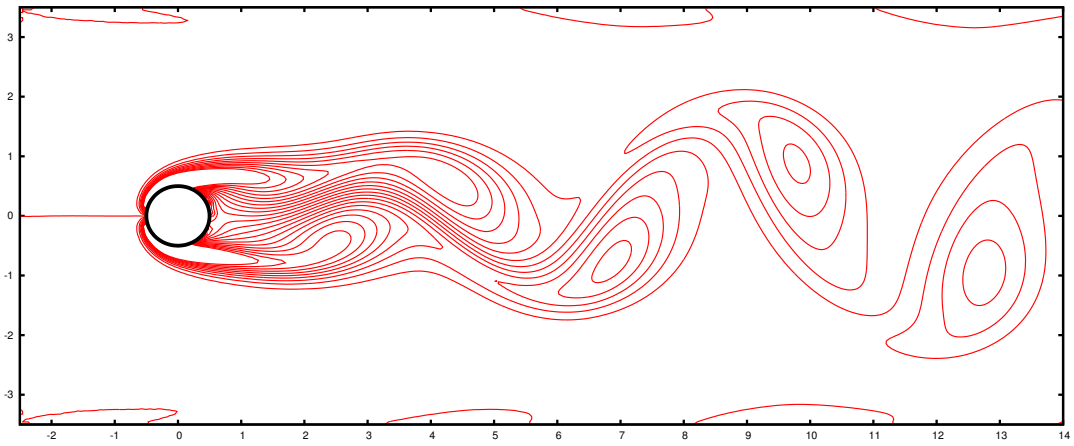
The Chorin splitting implementation at resolution 640×640 is taken to be the exact solution. The example was then run using the proposed method at resolutions 40×40, 80×80, 160×160, and 320×320. The differences for each velocity component are plotted in Figure 5.6(c) and Figure 5.6(d). The convergence order for $\mathbf{u}_1$ is 0.89, and the convergence order for $\mathbf{u}_2$ is 1.07. The final velocity profile is shown in Figure 5.6(b).

5.11.4 Vortex shedding

In this example we demonstrate Kármán vortex shedding around a cylinder and compare the dynamic behavior of our method in the case of low viscosity and one-way coupling against the results that others have obtained and against our implementation of



(a) Proposed method



(b) Chorin splitting

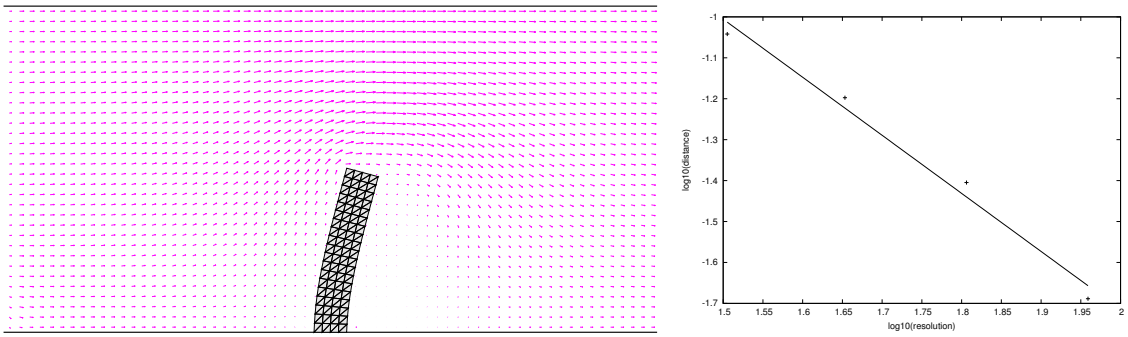
Figure 5.7: Vorticity contours demonstrating vortex shedding. Contours are provided for the range $[-2.5, 2.5]$ in increments of 0.25.

Chorin splitting. Our setup is based on that of [71, 29], except that we employ slightly different boundary conditions. Our domain is $[-2.5\,m, 15.0\,m] \times [-3.5\,m, 3.5\,m]$ with a fixed cylinder of radius $0.5\,m$ centered at the origin. The top and bottom of the domain walls have slip boundary conditions. The left boundary has a source term $\mathbf{u}_1 = 1\,ms^{-1}$ with a slipping boundary condition for $\mathbf{u}_2$. The right boundary has the boundary condition $p = 0$. The flow has $Re = 100$ with $\rho = 1kg\,m^{-2}$ and $\mu = 0.01\,kg\,s^{-1}$. We used a 400×160 grid for our proposed method and for Chorin

splitting. Our vorticity contours for our proposed method and Chorin splitting are shown in Figure 5.7. Our vorticity contours exhibit the same characteristics as those shown in [71, 29] as well as being quite similar to each other. We note that the contour plot for the proposed method has an additional zero contour near the left side. The vorticity in this region is quite small, reaching its peak at roughly 1% of the separation between two contour lines. At resolution 200×80, the vorticity peaks at roughly 3% of the separation between two contour lines, indicating that the extra contour line will go away under refinement. For the proposed method, the frequency and Strouhal number are $f = 0.160\,Hz$ and $St = 0.160$. For the Chorin splitting implementation, $f = 0.166\,Hz$ and $St = 0.166$. These compare favorably with [29], which reported $St = 0.164 - 0.173$ depending on the details of their mesh.

5.11.5 Flow past deformable beam

This problem deals with a deformable beam immersed in fluid, similar to problems investigated by [14, 175, 164, 165, 44, 133, 79, 176]. The fluid domain was $[0\,m, 2\,m] \times [0\,m, 1\,m]$ with velocity boundary conditions of $0.25\,ms^{-1}$ on the left side, slip walls on the top and bottom, and a $p = 0$ boundary condition on the right wall. The beam has no-slip boundary conditions. The fluid density is $100\,kg\,m^{-2}$, and the coefficient of dynamic viscosity is $\mu = 1\,kg\,s^{-1}$. The structure is discretized as a triangle mesh using a mass-spring model with 20 particles in the vertical direction and 5 in the horizontal, each of mass $0.1\,kg$ except for the bottom row of particles, which are treated as infinitely massive. The mesh is initially rectangular, with extent from $[0.95\,m, 1.05\,m] \times [0\,m, 0.5\,m]$. Edge springs with Young's modulus $E = 166\,kg\,m^{-1}s^{-2}$ and overdamping fraction 2 along with altitude springs with Young's modulus $E = 400\,kg\,m^{-1}s^{-2}$ and overdamping fraction 2 were used as the structure constitutive model. (If considered in isolation, a spring can be made critically damped by choosing a specific damping constant. When we refer to an overdamping fraction of 2, we mean that the damping coefficient on the spring is twice the value that would result in a critically damped spring. Isolated springs with an overdamping fraction greater than



(a) Fluid velocities for flow over deforming beam

(b) Position error $\|\mathbf{x} - \mathbf{x}_{ex}\|$.

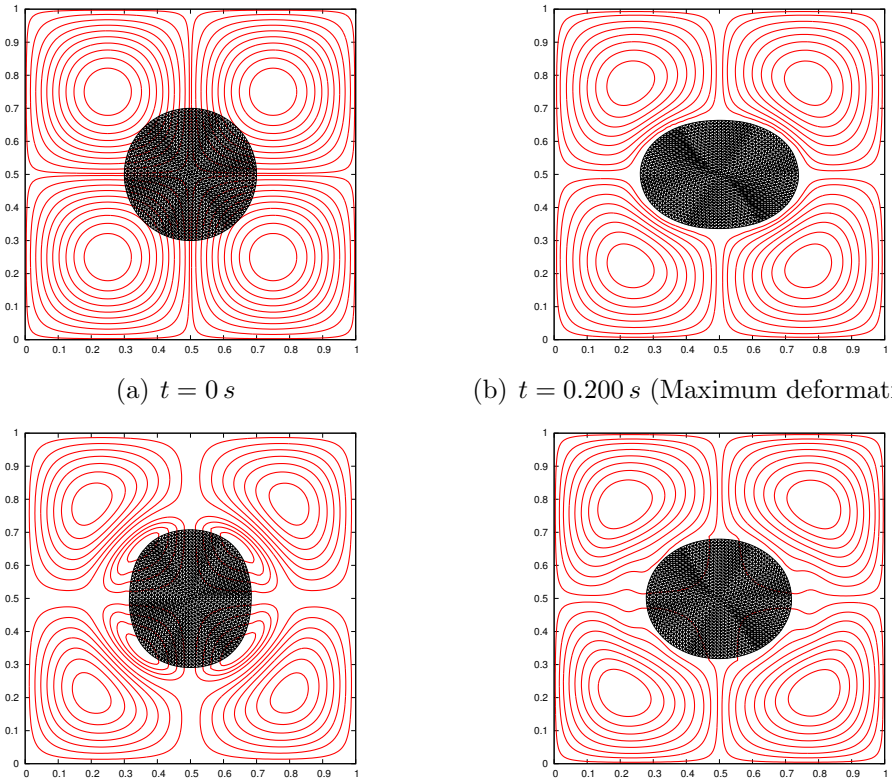
Figure 5.8: Velocity profile and position error are shown for the top right particle of the deforming beam at time 1 s. The position of the particle at the highest resolution is taken to be the exact value $\mathbf{x}_{ex}$.

one are overdamped, and isolated springs with an overdamping fraction less than one are underdamped. The analogy does not extend to systems of springs, but we find this approach to be more intuitive for describing the amount of damping present.) Altitude springs [102] create a spring between each particle of a triangle and a virtual node projected onto the opposite face and are used to give volumetric structure and prevent inversion.

Note for the purposes of our convergence analysis we take the structure to be a discrete system and refine only the fluid grid resolution. We examine convergence with fluid grid resolution and time step size in Figure 5.8(b). The error is computed assuming the simulation with resolution 256×128 is the exact solution. We plot the convergence of $\|\mathbf{x}-\mathbf{x}_{ex}\|$, where $\mathbf{x}_{ex}$ is the position in the exact solution configuration, for the top right particle of the beam at time 1 s in Figure 5.8(b). The convergence order is 1.42. The velocity profile is shown in Figure 5.8(a).

5.11.6 Oscillating disk

We have implemented the oscillating disk example from [176], where a circular deformable body is placed within a fluid domain of dimensions $[0\,m, 1\,m] \times [0\,m, 1\,m]$ with periodic boundary conditions and an initial velocity defined for both the fluid and



(a) $t = 0\,s$ (b) $t = 0.200\,s$ (Maximum deformation)

(c) $t = 0.500\,s$ (d) $t = 1.00\,s$

Figure 5.9: Streamlines in the oscillating disk example. The fluid grid resolution is 128×128 and the structure is composed of 2400 triangles.

the structure by the stream function $\psi = \psi_0 sin(k_x x) sin(k_y y)$, with $\psi_0 = 0.05 m^2 s^{-1}$ and $k_x = k_y = 2\pi m^{-1}$. We used a neo-Hookean constitutive model as described in [22] for the structure with Poisson's ratio $\nu = 0.475$, shear modulus $G = 1\,Pa$, and Rayleigh damping coefficient 0.005. The structure and fluid densities were set to $\rho = 1\,kg\,m^{-2}$, and the coefficient of dynamic viscosity was $\mu = 0.001\,kg\,s^{-1}$. We ran the simulation for a period of $1\,s$. For comparison to [176] we rasterize the structure velocities onto the fluid grid and display the streamlines and structure deformation (Figure 5.9). The frequency of the oscillator is similar to that in [176] as can be seen from the energy plots in Figure 5.10. We note that our simulation is more damped than theirs. Our extra damping is due to our use of semi-Lagrangian advection on the fluid and Rayleigh damping on the structure.

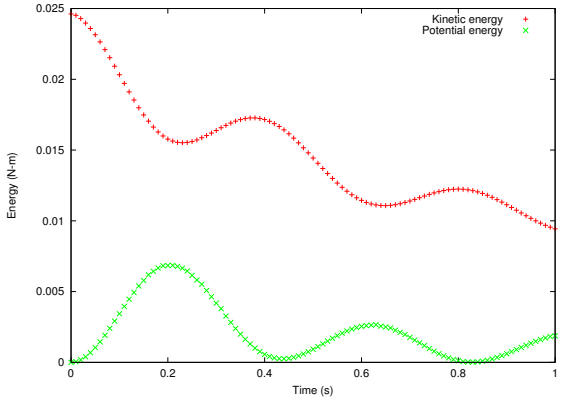


Figure 5.10: Energy over time for the oscillating disk shown in Figure 5.9.

5.12 Conclusions and Future Work

We have presented a method for formulating strongly coupled fluid structure interaction problems as a symmetric positive definite linear system. The method is based on factoring the structure damping matrix and performing a change of variables in the structure unknowns. It can be applied to arbitrary linearized structure constitutive models but requires a semi-implicit time discretization for the structure equations, where elastic terms are handled explicitly and damping terms are handled implicitly. For the special case of only rigid structures and no other rigid-rigid forces we recover exactly the method of [15]. The numerical experiments on problems common in the fluid structure interaction literature demonstrate that the method is convergent with grid resolution.

Second order accuracy for fluid structure interaction has been achieved by some authors, such as the formally second order methods [61, 60]. We note, however, that second order accuracy came well after the original method and that the additional accuracy comes at the cost of additional complexity. An approach which attempts to be second order accurate in space might resolve the boundary between the fluid and structure more accurately by placing constraints at the structure surface rather than at MAC face centers or otherwise take advantage of the available geometric information. Second order accuracy in time would be more difficult to achieve, as it

would require finding second order accurate integration schemes for the fluid and the structure that retain that accuracy when coupled. Instead of attempting to address second order accuracy in this paper, we leave the problem of higher order accuracy for future work.

Compared to standard Chorin splitting (without structures), our evolution will be significantly slower. Chorin splitting with implicit viscosity requires the solution of one well-conditioned viscosity system per dimension and a single poorly-conditioned Poisson system for which preconditioning has been well studied. We replace these with a single linear system that, due to the Poisson matrix it contains, is also poorly conditioned. We have observed its conditioning to be significantly worse at high viscosity than low viscosity, and we have also observed poor conditioning to result from underresolved regions. In particular, we have observed poor convergence when the structure is much coarser than the fluid grid and we add extra coupling faces to $\mathbf{W}$ as described in Section 5.7, possibly due to being overconstrained. Increasing the resolution of the structure improves convergence in these cases, but a more careful treatment of boundary conditions might avoid the issue entirely. Conditioning with deformable structures becomes worse with lower density and higher damping. The conditioning improves significantly when all structures are one-way coupled.

The system is symmetric positive definite, making it suitable for solution with methods like conjugate gradient and generally easier to precondition and solve than other types of linear systems. The solution of this system is still expensive in our implementation as we have not developed an effective preconditioner for it and have not made significant effort to optimize our routines. We leave the problem of finding a good preconditioner for future work. Solving the linear systems dominates the cost of our time evolution. The algorithm compares favorably with [125], which has an indefinite system that we have observed to converge much slower in practice.

Chapter 6

Implicit Surface Tension

In this chapter, we build a framework for applying implicit forces on Lagrangian degrees of freedom to an Eulerian discretization of the Navier-Stokes equations. We then leverage the framework established in Chapter 5 to produce a system that is symmetric and positive definite. We then use this framework to address the tight stability time step restriction imposed by surface tension. The work in this chapter is based on the work under review [128].

6.1 Introduction

Surface tension is a significant force in many phenomena, generally becoming more significant as features decrease in size. It plays a vital role in the dynamics of water droplets and bubbles, where the shape becomes increasingly dominated by surface tension forces as the size decreases. The foundations of surface tension simulation began with smeared δ-functions applied to interfaces maintained by front tracking [159], volume of fluid [26], and level sets [31, 146]. Another interesting early work on surface tension is [75]. Since then, a wide variety of different formulations have been investigated, and the difficulties posed by surface tension have been studied and

increasingly understood.

Surface tension is subject to a tight $\Delta t = O(\Delta x^{3/2})$ time step step restriction, and a significant number number of approaches have been considered for alleviating it. The early approach of [72] started from a CSF formulation and worked out how the curvature changes in time to derive an implicit formulation of surface tension. This was followed up by [77], which used a simplified formulation that also added additional damping along the interface. The approaches of [141, 172, 177] (see also [45]) improve stability by adding and subtracting a Laplacian, effectively introducing additional viscosity to the fluid. There has also been work on improving stability with VOF-based surface tension schemes [145, 99].

The tendency for surface tension to produce parasitic currents is well known in the literature. [83] aimed to eliminate parasitic currents in the second gradient formulation of surface tension at the cost of momentum conservation. Methods based on delta functions require careful attention to produce accurate results, as even $O(1)$ errors may result if not implemented carefully, a point which has been noted for example in the context of length computation [142, 46, 155]. In [86], it was noted that parasitic currents when using a regularized delta function formulation were found to be three orders of higher than were obtained from a formulation handling surface tension as a pressure jump across a sharp interface using the ghost fluid method. [54] showed that enforcing a balance between the pressure and the surface tension in a manner similar to the ghost fluid method was important. [55] also found that balancing the pressure with surface tension forces resulted in a more accurate formulation. [62], based on [63], proposed using XFEM to formulate a discontinuous pressure jump as well. See also [33, 34, 49]. Following these observations, we take a sharp-interface approach to applying surface tension, addressing accuracy through careful attention to the force balance between surface tension and pressure.

We develop a method for implicit surface tension based on a discretizing the Laplace Beltrami operator on a Lagrangian mesh, which we couple to an otherwise Eulerian discretization. The resulting scheme is very similar in form to CSF schemes, but we

are able to take advantage of the flexibility in mapping between degrees of freedom to reduce the severity of parasitic currents to levels comparable to those in [86]. While we formulate surface tension as a force, our surface tension discretization can also be formulated as a pressure jump.

We explore two approaches to maintaining our interface: level sets and front tracking. For a good review on front tracking, see [156]. Front tracking has the advantage that it is relatively easy to treat implicitly and is conveniently compatible with our treatment of surface tension through a Lagrangian surface mesh. For other examples of implicit front tracking see [96], which evolves an elastic membrane implicitly using and Jacobian-free Newton Krylov method solver, and [147], which uses the Jacobian-free Newton Krylov method to implicitly update control points.

6.2 Eulerian Navier Stokes with Lagrangian Forces

6.2.1 Navier Stokes Equations

In this paper, we consider the Eulerian form of the incompressible Navier Stokes equations,

$$\rho\left(\frac{\partial \vec{u}}{\partial t} + (\vec{u}\cdot\nabla)\vec{u}\right) = -\nabla p + \mu\nabla^2\vec{u} + \vec{f_\rho} \tag{6.1}$$

subject to the incompressibility constraint

$$\nabla\cdot\vec{u} = 0. \tag{6.2}$$

Here, $\vec{u}$ is the fluid velocity, p is the pressure, μ is the dynamic viscosity, ρ is the density, and $\mathbf{f}_\rho$ is an additional force density. Later, $\mathbf{f}_\rho$ will be a surface tension force.

Our spatial discretization is based on a standard Marker and Cell (MAC, [68]) grid discretization. Chorin splitting [37] is used to separate the advection term and explicit forces from the pressure and implicit forces. We apply advection to obtain an

intermediate velocity

$$\rho\vec{u}^{\star} = \rho\vec{u}^{n} - \Delta t\rho(\vec{u}\cdot\nabla)\vec{u} + \Delta t\vec{f}_{\rho 1}, \tag{6.3}$$

which is applied with 3rd order Runge Kutta [135] and 3rd order Hamilton Jacobi ENO [116]. The pressure and implicit forces are discretized as

$$\rho\vec{u}^{n+1} = \rho\vec{u}^{\star} - \Delta t\nabla p + \Delta t\mu\nabla^2\vec{u}^{n+1} + \Delta t\vec{f}_{\rho 2}. \tag{6.4}$$

It will be convenient to work in a fully-discretized and volume-weighted form, which is accomplished by first multiplying through by the cell volume V. The equations now take the form

$$\boldsymbol{\beta}\mathbf{u}^{n+1} = \boldsymbol{\beta}\mathbf{u}^{\star} - \Delta t\mathbf{G}\mathbf{p} - \Delta tV^{-1}\mu\hat{\mathbf{G}}_{\mu}^{T}\hat{\mathbf{G}}_{\mu}\mathbf{u}^{n+1} + \Delta t\mathbf{f}.$$

Here, $\mathbf{u}$ represents a vector of velocity degrees of freedom, $\boldsymbol{\beta} = V\rho$ is a diagonal matrix whose entries are the lumped fluid masses at faces, $\mathbf{G} = V\nabla$ is the discretized, volume-weighted pressure gradient, $\hat{\mathbf{G}}_{\mu} = V\nabla$ is the discretized, volume-weighted velocity gradient, $\mathbf{p}$ the vector of pressures, and $\mathbf{f}$ is the vector of additional implicit forces $V\vec{f}_{\rho 2}$. Finally, dividing by $\boldsymbol{\beta}$ and letting $\mathbf{G}_{\mu} = \sqrt{\Delta tV^{-1}\mu}\hat{\mathbf{G}}_{\mu}$ and $\hat{\mathbf{p}} = \Delta t\mathbf{p}$ for notational convenience,

$$\mathbf{u}^{n+1} = \mathbf{u}^{\star} - \boldsymbol{\beta}^{-1}\mathbf{G}\hat{\mathbf{p}} - \boldsymbol{\beta}^{-1}\mathbf{G}_{\mu}^{T}\mathbf{G}_{\mu}\mathbf{u}^{n+1} + \Delta t\boldsymbol{\beta}^{-1}\mathbf{f}. \tag{6.5}$$

The corresponding incompressibility constraint is

$$-\mathbf{G}^{T}\mathbf{u}^{n+1} = \mathbf{0}. \tag{6.6}$$

6.2.2 Structure Equations

Although an Eulerian framework is used for evolution, occasional references will be made of the evolution equations for structures as well. We use a traditional Lagrangian approach for structures, where we solve the equations $\dot{\mathbf{x}} = \mathbf{v}$ and $\mathbf{M}\dot{\mathbf{v}} = \hat{\mathbf{f}}$. Here, the quantities $\mathbf{x}$ and $\mathbf{v}$ are the vectors containing the position and velocity degrees of freedom for all of the Lagrangian particles, and $\mathbf{M}$ is the mass matrix. It suffices for our purposes to consider elastic forces, where the force $\hat{\mathbf{f}}$ is defined in terms of a potential energy as $\hat{\mathbf{f}} = -\frac{\partial \phi}{\partial \mathbf{x}}$, where ϕ is a potential energy function that is a function of only the positions $\mathbf{x}$.

One particularly simple and stable way of evolving these equations is via backward Euler. In this case, one solves

$$\begin{aligned}\mathbf{x}^{n+1} &= \mathbf{x}^n + \Delta t \mathbf{v}^{n+1} \\ \mathbf{v}^{n+1} &= \mathbf{v}^n + \Delta t \mathbf{M}^{-1} \hat{\mathbf{f}}^{n+1}\end{aligned}$$

taking the first order approximation $\hat{\mathbf{f}}^{n+1} = \hat{\mathbf{f}}^n + \frac{\partial \hat{\mathbf{f}}^n}{\partial \mathbf{x}}(\mathbf{x}^{n+1} - \mathbf{x}^n) + \frac{\partial \hat{\mathbf{f}}^n}{\partial \mathbf{v}}(\mathbf{v}^{n+1} - \mathbf{v}^n) = \hat{\mathbf{f}}^n + \Delta t \frac{\partial \hat{\mathbf{f}}^n}{\partial \mathbf{x}} \mathbf{v}^{n+1} = \hat{\mathbf{f}}_e + \mathbf{D}\mathbf{v}^{n+1}$, since $\hat{\mathbf{f}}^{n+1}$ does not have explicit velocity dependence. $\mathbf{f}_e$ and $\mathbf{D}\mathbf{v}^{n+1}$ represent the explicit and implicit portions of the force, respectively. The backward Euler update now takes the form

$$\begin{aligned}\mathbf{x}^{n+1} &= \mathbf{x}^n + \Delta t \mathbf{v}^{n+1} \\ \mathbf{v}^{n+1} &= \mathbf{v}^n + \Delta t \mathbf{M}^{-1} (\hat{\mathbf{f}}_e + \mathbf{D}\mathbf{v}^{n+1})\end{aligned} \tag{6.7}$$

$\mathbf{D} = \Delta t \frac{\partial \hat{\mathbf{f}}^n}{\partial \mathbf{x}} = -\frac{\partial^2 \phi}{\partial \mathbf{x} \partial \mathbf{x}}$ is symmetric since it is a second derivative. In practice, $\mathbf{D}$ will often be negative definite. This is the case for the surface tension force considered in this paper. Note that $\mathbf{D}$ contains the factor Δt, so that the damping forces due to the implicit force treatment vanish under temporal refinement.

6.2.3 Framework for Forces Discretized on a Lagrangian Mesh

Consider a single particle whose motion is computed from the velocities stored on an Eulerian grid. This particle does not have any real degrees of freedom, and it does not have mass. In general, these particles may be on the surface of a fluid region or in the interior. The Lagrangian velocities $\mathbf{v}$, a vector containing the velocities for all Lagrangian particles, are defined in terms of the vector of all Eulerian velocities $\mathbf{u}$ by the relation

$$\mathbf{v} = \mathbf{H}\mathbf{u}, \tag{6.8}$$

where $\mathbf{H}$ is some linear operator that interpolates Eulerian velocities to Lagrangian velocities.

The force $\mathbf{f}$ in (6.5) is discretized over the Eulerian degrees of freedom. We would like to instead add a force $\hat{\mathbf{f}}$ which is discretized over the new Lagrangian degrees of freedom. Let P be the power (work per unit time) performed by applying a force $\mathbf{f}$ to fluid moving with velocity $\mathbf{u}$. That is, $P = \mathbf{f}^T\mathbf{u}$. In this case, we computed the power using the Eulerian degrees of freedom. Computing it instead from the Lagrangian degrees of freedom produces $P = \hat{\mathbf{f}}^T\mathbf{v} = \hat{\mathbf{f}}^T\mathbf{H}\mathbf{u} = \mathbf{f}^T\mathbf{u}$. Since $\mathbf{v}$ and $\mathbf{f}$ could have been arbitrary, it follows that

$$\mathbf{f} = \mathbf{H}^T\hat{\mathbf{f}}.$$

Two variables whose inner product produces the work (or a related quantity like power) are said to be work conjugates. The above derivation can be considered an application of the principle of virtual work, a powerful tool that is describe in more detail in [22]. The operator $\mathbf{H}^T$ is a conservative force distribution operator. In general, whenever $\mathbf{H}$ is an interpolation operator from one set of degrees of freedom to another, the operator $\mathbf{H}^T$ can be used to distribute forces from the second set of degrees of freedom to the first.

Since the motion of the Lagrangian degrees of freedom is dictated by fluid motion, one could compute an *effective* mass for the Lagrangian particles. Observe that the velocity change of Eulerian degrees of freedom $\Delta\mathbf{u}$ due to the Lagrangian force is

$\Delta\mathbf{u} = \Delta t\boldsymbol{\beta}^{-1}\mathbf{f} = \Delta t\boldsymbol{\beta}^{-1}\mathbf{H}^T\hat{\mathbf{f}}$. The velocity change of Lagrangian degrees of freedom $\Delta\mathbf{v}$ due to the Lagrangian force is $\Delta\mathbf{v} = \mathbf{H}\Delta\mathbf{u} = \Delta t\mathbf{H}\boldsymbol{\beta}^{-1}\mathbf{H}^T\hat{\mathbf{f}} = \Delta t\mathbf{M}^{-1}\hat{\mathbf{f}}$, so that $\mathbf{M}^{-1} = \mathbf{H}\boldsymbol{\beta}^{-1}\mathbf{H}^T$ is the effective mass of the Lagrangian degrees of freedom.

In constitutive mechanics, Lagrangian forces for constitutive models have elastic and damping components. These components are typically assumed to take the form $\hat{\mathbf{f}} = \hat{\mathbf{f}}_e + \mathbf{D}\mathbf{v}$ [22]. Here, $\hat{\mathbf{f}}_e$ is the explicit force, and $\mathbf{D}$ is the symmetric negative semidefinite damping matrix. The implicit portion of the Navier Stokes equations now takes on the form

$$\mathbf{u}^{n+1} = \mathbf{u}^\star - \boldsymbol{\beta}^{-1}\mathbf{G}\hat{\mathbf{p}} - \boldsymbol{\beta}^{-1}\mathbf{G}_\mu^T\mathbf{G}_\mu\mathbf{u}^{n+1} + \Delta t\boldsymbol{\beta}^{-1}\mathbf{H}^T\hat{\mathbf{f}}_e + \Delta t\boldsymbol{\beta}^{-1}\mathbf{H}^T\mathbf{D}\mathbf{H}\mathbf{u}^{n+1}. \quad (6.9)$$

Observe that the implicit component of the additional force has the same form as the viscosity term. The explicit portion of the surface tension force goes in $\vec{f}_{\rho 1}$, and the implicit part goes in $\vec{f}_{\rho 2}$. (6.3) and (6.4) can be written (in fully discretized form) as

$$\mathbf{u}^\star = \mathbf{u}^n - \Delta t\mathbf{A}(\mathbf{u}) + \Delta t\boldsymbol{\beta}^{-1}\mathbf{H}^T\hat{\mathbf{f}}_e \quad (6.10)$$

$$\mathbf{u}^{n+1} = \mathbf{u}^\star - \boldsymbol{\beta}^{-1}\mathbf{G}\hat{\mathbf{p}} - \boldsymbol{\beta}^{-1}\mathbf{G}_\mu^T\mathbf{G}_\mu\mathbf{u}^{n+1} + \Delta t\boldsymbol{\beta}^{-1}\mathbf{H}^T\mathbf{D}\mathbf{H}\mathbf{u}^{n+1}, \quad (6.11)$$

where $\mathbf{A}(\mathbf{u})$ is the discretized advective term. We have effectively constructed a methodology for formulating arbitrary forces on a set of Lagrangian degrees of freedom in a way that results in a purely Eulerian problem and such that the linearized force terms take on the same form as the viscosity term. The framework of [124] allows one to construct a symmetric and positive definite system where viscosity is applied implicitly and coupled to the pressure projection. Since we have expressed the implicit portion of the Lagrangian forces in the same form, we can implicitly compute the pressure, viscosity, and Lagrangian forces in a way that is fully coupled and produces a symmetric and positive definite system. We assemble this system in Section 6.2.6.

This combination is particularly appealing in the case of surface tension. Both surface tension forces and pressure forces are stiff, and they are frequently in a delicate balance. This point is exemplified by the classical and important situation of a

stationary circle, which is considered in Section 6.4.2. If surface tension is treated explicitly or in an implicit step separate from pressure, the surface tension force and the pressure force will be only weak coupling. The proposed method, by contrast, produces strong coupling between surface tension and pressure.

6.2.4 Fluid-Structure Interaction Equations

The derivation in Section 6.2.3 for applying forces on a Lagrangian mesh to an Eulerian simulation can also be obtained directly by considering the case of fluid-structure interaction where the Lagrangian forces are applied to a Lagrangian structure which is in turn coupled to an Eulerian fluid. The equations for fluid-structure interaction are constructed from the equations of fluid dynamics and structure mechanics by the addition of a contact constraint, which is enforced through a momentum exchange. The contact constraint ensures that the relative velocity between fluid and structure is zero at a set of constraint locations. In Chapter 5 (see also [124]) these constraint locations were placed at fluid faces near the fluid-structure interface. The contact constraint can be written $\mathbf{J}\mathbf{v}^{n+1} = \mathbf{W}\mathbf{u}^{n+1}$, where $\mathbf{J}$ is the matrix that interpolates structure velocities to constraint locations and $\mathbf{W}$ is the matrix that interpolates fluid velocities to constraint locations. Note that this convention differs slightly from that in Chapter 5, where $\mathbf{J}$ is replaced by $\mathbf{WJ}$. The constraint is enforced by exchanging impulses $\boldsymbol{\lambda}$ between the structure and the fluid at the constraint locations. As noted in Section 6.2.3, since fluid velocities are interpolated to the constraint locations with $\mathbf{W}$, the impulses $\boldsymbol{\lambda}$ should be applied back to the fluid using $\mathbf{W}^T$. This produces the modified fluid equation,

$$\boldsymbol{\beta}\mathbf{u}^{n+1} = \boldsymbol{\beta}\mathbf{u}^{\star} - \mathbf{G}\hat{\mathbf{p}} + \mathbf{W}^T\boldsymbol{\lambda},$$

where for simplicity the viscosity term is ignored. An equal and opposite impulse is applied to the structure similarly using $\mathbf{J}^T$. The modified structure equation is

$$\mathbf{M}\mathbf{v}^{n+1} = \mathbf{M}\mathbf{v}^{\star} + \Delta t\hat{\mathbf{f}}_e + \Delta t\mathbf{D}\mathbf{v}^{n+1} - \mathbf{J}^T\boldsymbol{\lambda}.$$

Finally the fluid-structure interaction equations are

$$\boldsymbol{\beta}\mathbf{u}^{n+1} = \boldsymbol{\beta}\mathbf{u}^{\star} - \mathbf{G}\hat{\mathbf{p}} + \mathbf{W}^T\boldsymbol{\lambda} \tag{6.12}$$

$$\mathbf{M}\mathbf{v}^{n+1} = \mathbf{M}\mathbf{v}^{\star} + \Delta t\hat{\mathbf{f}} + \Delta t\mathbf{D}\mathbf{v}^{n+1} - \mathbf{J}^T\boldsymbol{\lambda} \tag{6.13}$$

$$\mathbf{G}^T\mathbf{u}^{n+1} = \mathbf{0} \tag{6.14}$$

$$\mathbf{J}\mathbf{v}^{n+1} = \mathbf{W}\mathbf{u}^{n+1} \tag{6.15}$$

For more details see Chapter 5.

6.2.5 Derivation from Fluid Structure Interaction

Consider the FSI coupling equations in Section 6.2.4, if $\mathbf{J}$ was an invertible matrix, it would be possible to eliminate $\mathbf{v}^{n+1}$ from (6.13) by solving (6.15). Normally, however, $\mathbf{J}$ will not be an invertible matrix. In this case, it is still possible to express $\mathbf{v}^{n+1}$ as the sum of two vectors, one ($\mathbf{J}^T\boldsymbol{\psi}$) in the range of $\mathbf{J}^T$ and one ($\hat{\mathbf{v}}$) orthogonal to it. There will in general be many suitable choices for $\boldsymbol{\psi}$, since $\boldsymbol{\psi}$ could contain components in the nullspace of $\mathbf{J}^T$. The vector $\boldsymbol{\psi}$ is selected to be the unique solution with minimum norm. The quantity $\boldsymbol{\psi}$ is defined at the constraint degrees of freedom. With this, $\mathbf{v}^{n+1}$ can be written as

$$\mathbf{v}^{n+1} = \mathbf{J}^T\boldsymbol{\psi} + \hat{\mathbf{v}} \qquad \mathbf{J}\hat{\mathbf{v}} = \mathbf{0}. \tag{6.16}$$

The vector $\boldsymbol{\psi}$ can now be eliminated. Substituting (6.16) into (6.15) produces $\mathbf{J}\mathbf{J}^T\boldsymbol{\psi} + \mathbf{J}\hat{\mathbf{v}} = \mathbf{W}\mathbf{u}^{n+1}$, which simplifies to

$$\mathbf{J}\mathbf{J}^T\boldsymbol{\psi} = \mathbf{W}\mathbf{u}^{n+1} \tag{6.17}$$

using (6.16). The minimum norm solution to (6.17) is obtained by taking the pseudoinverse, denoted by $\cdot^+$.

$$\boldsymbol{\psi} = (\mathbf{J}\mathbf{J}^T)^+\mathbf{W}\mathbf{u}^{n+1} \tag{6.18}$$

Substituting (6.18) into (6.16) produces

$$\mathbf{v}^{n+1} = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^+\mathbf{W}\mathbf{u}^{n+1} + \hat{\mathbf{v}} = \mathbf{H}\mathbf{u}^{n+1} + \hat{\mathbf{v}}, \tag{6.19}$$

where the definition

$$\mathbf{H} = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^+\mathbf{W} \tag{6.20}$$

has been used. The operator $\mathbf{H}$ maps fluid velocities to solid velocities and behaves as an interpolation operator. The operator $\mathbf{W}$ interpolates fluid velocities to constraint degrees of freedom, and $\mathbf{J}$ interpolates solid velocities to constraint degrees of freedom. The operator $\mathbf{J}^T(\mathbf{J}\mathbf{J}^T)^+$ is almost the inverse of the interpolation operator $\mathbf{J}$. Indeed, $(\mathbf{J}\mathbf{J}^T)(\mathbf{J}\mathbf{J}^T)^+$ is as nearly an identity matrix as is possible.

Next, we decompose $\boldsymbol{\lambda}$ using a similar orthogonal decomposition

$$\boldsymbol{\lambda} = \mathbf{J}\boldsymbol{\xi} + \hat{\boldsymbol{\lambda}} \qquad \mathbf{J}^T\hat{\boldsymbol{\lambda}} = \mathbf{0}, \tag{6.21}$$

where this time $\boldsymbol{\xi}$ is located at Lagrangian velocity degrees of freedom. As before, $\boldsymbol{\xi}$ is taken to be of minimum norm, so that it is uniquely determined. Next, $\boldsymbol{\xi}$ is eliminated. Substituting (6.21) into (6.13) produces

$$\mathbf{J}^T\mathbf{J}\boldsymbol{\xi} + \mathbf{J}^T\hat{\boldsymbol{\lambda}} = (\Delta t\hat{\mathbf{f}} + \mathbf{M}\mathbf{v}^\star) + (\Delta t\mathbf{D} - \mathbf{M})\mathbf{v}^{n+1},$$

which can be further simplified and solved using the pseudoinverse to produce

$$\boldsymbol{\xi} = (\mathbf{J}^T\mathbf{J})^+((\Delta t\hat{\mathbf{f}} + \mathbf{M}\mathbf{v}^\star) + (\Delta t\mathbf{D} - \mathbf{M})\mathbf{v}^{n+1}). \tag{6.22}$$

Substituting (6.22) into (6.21) produces

$$\boldsymbol{\lambda} = \mathbf{J}(\mathbf{J}^T\mathbf{J})^+((\Delta t\hat{\mathbf{f}} + \mathbf{M}\mathbf{v}^\star) + (\Delta t\mathbf{D} - \mathbf{M})\mathbf{v}^{n+1}) + \hat{\boldsymbol{\lambda}}.$$

Applying the identity $\mathbf{J}(\mathbf{J}^T\mathbf{J})^+ = (\mathbf{J}\mathbf{J}^T)^+\mathbf{J}$ produces

$$\boldsymbol{\lambda} = (\mathbf{J}\mathbf{J}^T)^+\mathbf{J}((\Delta t\hat{\mathbf{f}} + \mathbf{M}\mathbf{v}^\star) + (\Delta t\mathbf{D} - \mathbf{M})\mathbf{v}^{n+1}) + \hat{\boldsymbol{\lambda}}. \tag{6.23}$$

Substituting (6.23) into $\mathbf{W}^T\boldsymbol{\lambda}$ and simplifying with (6.20) produces

$$\mathbf{W}^T\boldsymbol{\lambda} = \mathbf{H}^T((\Delta t\hat{\mathbf{f}} + \mathbf{M}\mathbf{v}^\star) + (\Delta t\mathbf{D} - \mathbf{M})\mathbf{v}^{n+1}) + \mathbf{W}^T\hat{\boldsymbol{\lambda}}. \tag{6.24}$$

Finally, substitute (6.24) into (6.12),

$$\boldsymbol{\beta}\mathbf{u}^{n+1} = \boldsymbol{\beta}\mathbf{u}^\star - \mathbf{G}\hat{\mathbf{p}} + \mathbf{H}^T((\Delta t\hat{\mathbf{f}} + \mathbf{M}\mathbf{v}^\star) + (\Delta t\mathbf{D} - \mathbf{M})\mathbf{v}^{n+1}) + \mathbf{W}^T\hat{\boldsymbol{\lambda}}.$$

Rearranging and converting from Lagrangian degrees of freedom to the Eulerian degrees of freedom using (6.19) produces

$$\boldsymbol{\beta}\mathbf{u}^{n+1} = \boldsymbol{\beta}\mathbf{u}^\star - \mathbf{G}\hat{\mathbf{p}} + \mathbf{H}^T(\Delta t\mathbf{D} - \mathbf{M})\mathbf{H}\mathbf{u}^{n+1} + \mathbf{H}^T(\Delta t\hat{\mathbf{f}} + \mathbf{M}\mathbf{v}^\star) + \mathbf{H}^T(\Delta t\mathbf{D} - \mathbf{M})\hat{\mathbf{v}} + \mathbf{W}^T\hat{\boldsymbol{\lambda}}.$$

The quantities $\hat{\mathbf{v}}$ and $\hat{\boldsymbol{\lambda}}$ are problematic, and it is desirable to choose a discretization so that they vanish. One way of doing this is to introduce one (independent) coupling constraint per solid degree of freedom. Then $\mathbf{J}$ is an invertible matrix, and $\hat{\mathbf{v}}$ and $\hat{\boldsymbol{\lambda}}$ vanish. This produces the lumped mass formulation

$$\boldsymbol{\beta}\mathbf{u}^{n+1} = \boldsymbol{\beta}\mathbf{u}^\star - \mathbf{G}\hat{\mathbf{p}} + \mathbf{H}^T(\Delta t\mathbf{D} - \mathbf{M})\mathbf{H}\mathbf{u}^{n+1} + \mathbf{H}^T(\Delta t\hat{\mathbf{f}} + \mathbf{M}\mathbf{v}^\star)$$

This can be placed into a particularly appealing form using the simplifying assumption $\mathbf{v}^\star = \mathbf{H}\mathbf{u}^\star$ and the lumped mass $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta} + \mathbf{H}^T\mathbf{M}\mathbf{H}$

$$\hat{\boldsymbol{\beta}}\mathbf{u}^{n+1} = \boldsymbol{\beta}\mathbf{u}^\star + \mathbf{H}^T\mathbf{M}\mathbf{v}^\star - \mathbf{G}\hat{\mathbf{p}} + \mathbf{H}^T\Delta t\hat{\mathbf{f}} + \Delta t\mathbf{H}^T\mathbf{D}\mathbf{H}\mathbf{u}^{n+1} \tag{6.25}$$

In the limit of a massless structure, $\mathbf{M} = \mathbf{0}$, and $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}$, so that what remains is the massless formulation

$$\boldsymbol{\beta}\mathbf{u}^{n+1} = \boldsymbol{\beta}\mathbf{u}^\star - \mathbf{G}\hat{\mathbf{p}} + \Delta t\mathbf{H}^T\hat{\mathbf{f}} + \Delta t\mathbf{H}^T\mathbf{D}\mathbf{H}\mathbf{u}^{n+1}. \tag{6.26}$$

This equation depends only on the ability to eliminate the velocity degrees of freedom using the contact constraints.

In summary, we have shown that if one there is one constraint per Lagrangian degree of freedom, so that $\mathbf{J}$ is invertible, then the FSI problem in (6.12) through (6.15) can be entirely written in terms of the Eulerian degrees of freedom on the Eulerian mesh with the momentum in $\mathbf{v}^\star$ transfered to fluid with $\mathbf{H}^T$. What remains is an Eulerian problem to solve. (6.15) is unchanged, but the other three FSI equations are replaced with (6.25). In the event that the Lagrangian mesh is massless, as we propose, then we are left with (6.15) and (6.26). This provides legitimacy to the the last two terms in (6.9). We can now revert back to (6.6), (6.10), and (6.11), which also include viscosity, advection, and other forces.

6.2.6 SPD Linear System

A symmetric positive definite (SPD) system is formed from (6.11) and (6.6) following the approach of Chapter 5, which we utilize in a somewhat more concise way. If $\mathbf{D}$ is symmetric negative semidefinite, there will exist a matrix $\mathbf{C}$ such that $\mathbf{C}^T\mathbf{C} = -\Delta t\mathbf{D}$.

$$\mathbf{u}^{n+1} = \mathbf{u}^\star - \boldsymbol{\beta}^{-1}\mathbf{G}\hat{\mathbf{p}} - \boldsymbol{\beta}^{-1}\mathbf{G}_\mu^T\mathbf{G}_\mu\mathbf{u}^{n+1} - \boldsymbol{\beta}^{-1}\mathbf{H}^T\mathbf{C}^T\mathbf{C}\mathbf{H}\mathbf{u}^{n+1}. \tag{6.27}$$

Rewriting this using block matrices produces

$$\mathbf{u}^{n+1} = \mathbf{u}^\star - \boldsymbol{\beta}^{-1}\begin{pmatrix}\mathbf{G}^T\\ \mathbf{G}_\mu\\ \mathbf{C}\mathbf{H}\end{pmatrix}^T\begin{pmatrix}\hat{\mathbf{p}}\\ \mathbf{G}_\mu\mathbf{u}^{n+1}\\ \mathbf{C}\mathbf{H}\mathbf{u}^{n+1}\end{pmatrix}. \tag{6.28}$$

(6.28) can be written more concisely as

$$\mathbf{u}^{n+1} = \mathbf{u}^\star - \boldsymbol{\beta}^{-1}\mathbf{K}^T\mathbf{z}, \tag{6.29}$$

where

$$\mathbf{K} = \begin{pmatrix}\mathbf{G}^T\\ \mathbf{G}_\mu\\ \mathbf{C}\mathbf{H}\end{pmatrix} \qquad \mathbf{z} = \begin{pmatrix}\hat{\mathbf{p}}\\ \mathbf{G}_\mu\mathbf{u}^{n+1}\\ \mathbf{C}\mathbf{H}\mathbf{u}^{n+1}\end{pmatrix}. \tag{6.30}$$

Left multiplying (6.29) by $\mathbf{K}$ and rearranging produces

$$\mathbf{K}\mathbf{u}^{n+1} + \mathbf{K}\boldsymbol{\beta}^{-1}\mathbf{K}^T\mathbf{z} = \mathbf{K}\mathbf{u}^\star. \tag{6.31}$$

This can be simplified by using (6.6) and noting that

$$\mathbf{K}\mathbf{u}^{n+1} = \begin{pmatrix} \mathbf{G}^T \\ \mathbf{G}_\mu \\ \mathbf{C}\mathbf{H} \end{pmatrix} \mathbf{u}^{n+1} = \begin{pmatrix} \mathbf{0} \\ \mathbf{G}_\mu\mathbf{u}^{n+1} \\ \mathbf{C}\mathbf{H}\mathbf{u}^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{p}} \\ \mathbf{G}_\mu\mathbf{u}^{n+1} \\ \mathbf{C}\mathbf{H}\mathbf{u}^{n+1} \end{pmatrix} = \mathbf{P}\mathbf{z}, \tag{6.32}$$

where

$$\mathbf{P} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix}$$

and $\mathbf{I}$ is the identity matrix. Substituting (6.32) into (6.31) produces the final SPD system

$$(\mathbf{P} + \mathbf{K}\boldsymbol{\beta}^{-1}\mathbf{K}^T)\mathbf{z} = \mathbf{K}\mathbf{u}^\star. \tag{6.33}$$

This system is symmetric positive semidefinite since it is the sum of two symmetric positive semidefinite matrices. Note that to make this system SPD, the unknowns must be changed, and this requires that the damping forces factor. The viscosity factors in a straightforward way. For many force models, including the surface tension model proposed in this paper, the matrix $\mathbf{D}$ factors in a straightforward way as well. Solving this system produces $\mathbf{z}$, and the desired result $\mathbf{u}^{n+1}$ can be obtained by substituting into (6.29).

It is possible to characterize the nullspace of this system. To do this, assume $(\mathbf{P} + \mathbf{K}\boldsymbol{\beta}^{-1}\mathbf{K}^T)\mathbf{z} = \mathbf{0}$. Semidefiniteness of each part implies $\mathbf{P}\mathbf{z} = \mathbf{0}$ and $\mathbf{K}\boldsymbol{\beta}^{-1}\mathbf{K}^T\mathbf{z} = \mathbf{0}$. Since $\boldsymbol{\beta}^{-1}$ is nonsingular, it must be that $\mathbf{K}^T\mathbf{z} = \mathbf{0}$. The requirement $\mathbf{P}\mathbf{z} = \mathbf{0}$ leads to

$$\mathbf{z} = \begin{pmatrix} \hat{\mathbf{p}} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}.$$

The second requirement implies that $\mathbf{G}\hat{\mathbf{p}} = \mathbf{0}$, which may occur when all Neumann boundary conditions are imposed. That is, our new system will have a nullspace under the same circumstances as a one-phase incompressible simulation. Note that if $\mathbf{z}$ is a nullspace vector, then $\mathbf{z}^T\mathbf{K}\mathbf{u}^\star = \mathbf{0}$, so that (6.33) remains consistent even when the system is singular. That is, the right hand side is in the range of the system, so no projection of the right hand side is required.

6.2.7 Comments on Extensions to Hybrid Solids

The framework of [138] provides a means of applying forces at locations other than the locations of the actual degrees of freedom. It does this by introducing a notion of a hard bound particle, which is a particle whose position and velocity are defined implicitly as a linear combination of the actual degrees of freedom. This makes it possible, for example, to embed a detailed mesh for resolving collisions inside a coarser and better-conditioned simulation mesh. If $\mathbf{H}$ is the operator that interpolates from velocity degrees of freedom to the hard bound velocities then the force felt at velocity degrees of freedom due to forces on the hard bound particles is $\mathbf{H}^T\hat{\mathbf{f}}+\mathbf{H}^T\mathbf{D}\mathbf{H}\mathbf{v}$, where $\mathbf{v}$ are their Lagrangian velocity degrees of freedom. In each case, the force is applied by interpolating velocities from real degrees of freedom to extra degrees of freedom using $\mathbf{H}$, computing the force over the extra degrees of freedom, and then distributing forces back to the real degrees of freedom using $\mathbf{H}$. What is different in this case is that our real degrees of freedom are Eulerian rather than Lagrangian.

There is nothing special about defining Lagrangian degrees of freedom in terms of other degrees of freedom. Indeed, one could also define Eulerian velocity degrees of freedom in terms of other Eulerian degrees of freedom or in terms of Lagrangian degrees of freedom. One of the applications proposed in [138] for the purely Lagrangian form is to stitch together meshes with T-junctions. In the context of a purely Eulerian simulation, one could use it, for example, to simplify the discretization of adaptive methods. Thus, for an octree discretization, each refinement level could be discretized

independently. Then, where different levels meet, the coarse face velocities can be defined implicitly in terms of the fine velocity degrees of freedom. The interpolation in this case can simply be chosen so that the flux measured by the coarse or fine degrees of freedom agree. This effectively decouples the problem of operator discretization from that of managing degrees of freedom. Symmetry and definiteness are obtained very easily.

Similarly, one might consider defining Eulerian degrees of freedom implicitly in terms of Lagrangian degrees of freedom. One might, for example, treat the particles of a fluid implicit particle (FLIP) scheme as being the real degrees of freedom, with the Eulerian velocities required by the pressure projection step defined implicitly in terms of the particle degrees of freedom. In this way, the composition of the fluid interpolation operator and the divergence operator effectively defines a discretization of a Lagrangian divergence operator over the particle degrees of freedom.

6.3 Spatial Discretization

Up to this point, we have been working in discrete form but have only described the temporal discretization. In this section, we address our spatial discretization. Evolving the proposed scheme requires the discretization of $\boldsymbol{\beta}$, $\mathbf{G}$, $\mathbf{H}$, $\mathbf{C}$, and $\hat{\mathbf{G}}_\mu$. We consider both one-phase and two-phase discretizations. Because the extension to two-phase is straightforward, the one-phase case is considered, and differences in the two-phase case are discussed as appropriate. In the one-phase case, the fluid region is bounded by a free surface with a constant exterior Dirichlet pressure boundary condition and a stress-free Neumann velocity boundary condition.

In the two-phase case, the velocity field is assumed to be continuous across the interface. This simplifies advection and the discretization of $\hat{\mathbf{G}}_\mu$, neither of which are the focus of this paper. This assumption is not a limitation of the proposed technique, as the discretization of $\mathbf{G}$ very naturally handles a discontinuous velocity field, and the remaining operators do not depend on the velocity discretization.

Neither advection nor viscosity play an important role in the proposed algorithm, so their discretizations are not particularly important. Our advective term is discretized using 3rd order Hamilton Jacobi ENO [69, 135, 136]. For viscosity in the one-phase case, a standard central differencing discretization of $\hat{\mathbf{G}}_\mu$ is applied with a simple first order treatment of the Neumann boundary condition at the interface. In the two-phase case, the velocity field is continuous and standard central differencing is used throughout. The operators that remain to be discretized are $\boldsymbol{\beta}$ (Section 6.3.4), $\mathbf{G}$ (Section 6.3.4), $\mathbf{H}$ (Section 6.3.5), and $\mathbf{C}$ (Section 6.3.2).

6.3.1 Lagrangian Degrees of Freedom

In this paper a Lagrangian discretization is used for the purpose of computing surface tension, and therefore require a discretization of the surface. The details of the surface mesh are not very important, though for simplicity in discretization both the fluid grid and surface mesh are assumed to be fine enough to resolve the interface. Two different approaches to maintaining this surface discretization are considered, each suited to a different underlying interface representation. There many other possible approaches for representing the interface, including volume of fluid [26], interface functions [137], or meshless Lagrangian approaches such as moving least squares [58]. An alternative to classical front tracking is to instead track the entire fluid domain with a mesh, which leads naturally to ALE or fully Lagrangian approaches, as was done for example in [110].

Front Tracking

The first approach we consider is to use a front tracking surface mesh [156]. The positions of the interface particles are evolved using $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^{n+1}$, where $\mathbf{v}^{n+1} = \mathbf{H}\mathbf{u}^{n+1}$. In this case, the time evolution of the interface closely resembles backward Euler. If advection and forces on the Eulerian degrees of freedom are

ignored, the fluid update looks like

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \boldsymbol{\beta}^{-1} \mathbf{H}^T \hat{\mathbf{f}}_e + \Delta t \boldsymbol{\beta}^{-1} \mathbf{H}^T \mathbf{D} \mathbf{H} \mathbf{u}^{n+1}.$$

Multiplying by $\mathbf{H}$ produces

$$\mathbf{H}\mathbf{u}^{n+1} = \mathbf{H}\mathbf{u}^n + \Delta t \mathbf{H} \boldsymbol{\beta}^{-1} \mathbf{H}^T \hat{\mathbf{f}}_e + \Delta t \mathbf{H} \boldsymbol{\beta}^{-1} \mathbf{H}^T \mathbf{D} \mathbf{H} \mathbf{u}^{n+1}.$$

Let $\mathbf{M}^{-1} = \mathbf{H} \boldsymbol{\beta}^{-1} \mathbf{H}^T$ be the effective mass (inverse) of the structure. Making the approximation $\mathbf{v}^n = \mathbf{H}\mathbf{u}^n$ (this is not strictly true since $\mathbf{H}$ changes each time step as the structure moves through the Eulerian grid) produces

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta t \mathbf{M}^{-1} (\hat{\mathbf{f}}_e + \mathbf{D}\mathbf{v}^{n+1}) = \mathbf{v}^n + \Delta t \mathbf{M}^{-1} \hat{\mathbf{f}}^{n+1}.$$

This, along with $\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \mathbf{v}^{n+1}$, is a backward Euler update for a structure which has surface tension forces and an effective mass.

A level set is rebuilt from the front-tracked surface mesh after each time step to simplify the discretization of other operators. Since only inside-outside tests will be required of the level set in this case, the signed distance is computed from the surface mesh exactly for the band $|\phi| < \Delta x$ to ensure consistency between the two representations. The remainder of the level set is computed using fast marching [157, 158, 131, 70, 132, 36].

In order to maintain a valid and good quality surface mesh, remeshing is performed whenever necessary. The remeshing is triggered when any segment has length larger than $1.5\Delta x$ or smaller than $0.5\Delta x$, where Δx is the size of an Eulerian grid cell. Then a long segment will be split into two segments by adding a new point through a third order stencil, $-\frac{1}{16}\mathbf{x}_{k-1} + \frac{9}{16}\mathbf{x}_k + \frac{9}{16}\mathbf{x}_{k+1} - \frac{1}{16}\mathbf{x}_{k+2}$. Short segments will be merged into adjacent segments by deleting surface points until the resulting length is larger than the lower bound.

Level Set Tracked

The second approach considered for maintaining the surface mesh is to compute it from a level set representation of the interface at each time step. In this case, the particle level set method [47] is used to evolve the level set. The method used to compute the surface mesh from the level set is similar to marching cubes [100] but with careful attention paid to accuracy and surface mesh quality. Variations for computing surface meshes whose nodes are third and fourth order accurate are described. Since the surface mesh will be used to compute curvature, which is a second derivative, third order accuracy is required to obtain a first order accurate curvature. We use the fourth order discretization in all of our examples, but we demonstrate the accuracy of both variations in Section 6.4.1.

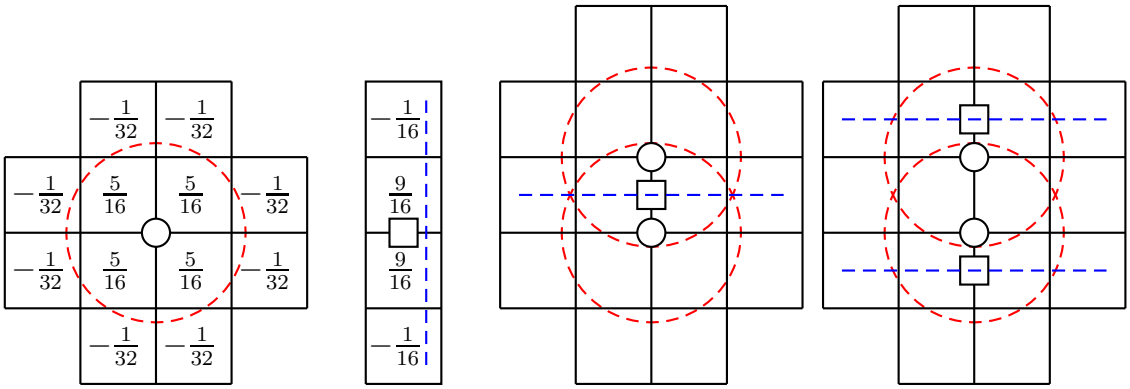
It is convenient to perform marching cubes such that the squares coincide with the fluid grid cells, since this will produce a mesh that is very convenient for the discretization of $\mathbf{G}$. Unfortunately, the fluid level set lives at cell centers and must be averaged to nodes. We use the standard fourth order stencil in Figure 6.1(a),

$$\begin{aligned}\phi_{j+\frac{1}{2},k+\frac{1}{2}} = &-\frac{1}{32}(\phi_{j-1,k} + \phi_{j+2,k} + \phi_{j-1,k+1} + \phi_{j+2,k+1}) \\ &- \frac{1}{32}(\phi_{j,k-1} + \phi_{j+1,k-1} + \phi_{j,k+2} + \phi_{j+1,k+2}) \\ &+ \frac{5}{16}(\phi_{j,k} + \phi_{j+1,k} + \phi_{j,k+1} + \phi_{j+1,k+1}).\end{aligned}$$

These node-centered level set values are used to determine the location of level set crossings and also as part of the computation of the location of those crossings.

The location of the level set crossing could be computed by constructing an interpolating polynomial from the node-centered level set. This results in an unnecessarily large stencil, which we would like to avoid. Instead one can interpolate the cell-centered level set to grid faces using the stencil in Figure 6.1(b)

$$\phi_{j+\frac{1}{2},k} = -\frac{1}{16}(\phi_{j-1,k} + \phi_{j+2,k}) + \frac{9}{16}(\phi_{j,k} + \phi_{j+1,k}).$$



(a) Node stencil (b) Face stencil (c) Third order stencil (d) Fourth order stencil

Figure 6.1: Level set samples are located at cell centers. These stencils are used to interpolate the level set to nodes (indicated with circles) and faces (indicated with squares) with the weights shown in the first two diagrams. The other two diagrams show stencils suitable for computing the location of a level set crossing along a face. The signs at the circled nodes are used to determine whether a crossing exists. The level set values averaged to the circled and squared locations are used to construct an interpolating polynomial. The level set crossing is taken to be the zero of this polynomial. The dashed lines and dashed circles indicate the cells used to compute each interpolated value.

to provide additional samples for computing an interpolation polynomial.

We provide an interpolating polynomial for third order and fourth order. The third order accurate crossing is obtained from the quadratic formed by interpolating the three points shown in Figure 6.1(c)

$$\left(0, \phi_{j+\frac{1}{2},k-\frac{1}{2}}\right) \qquad \left(\frac{1}{2}, \phi_{j+\frac{1}{2},k}\right) \qquad \left(1, \phi_{j+\frac{1}{2},k+\frac{1}{2}}\right).$$

The third order crossing is obtained by locating a root θ of the interpolating polynomial in the interval $[0, 1]$. The crossing occurs at the position $(1-\theta)\mathbf{x}_{j+\frac{1}{2},k-\frac{1}{2}} + \theta\mathbf{x}_{j+\frac{1}{2},k+\frac{1}{2}}$. The fourth order accurate crossing is a obtained from the cubic formed by interpolating the four points shown in Figure 6.1(d)

$$\left(-\frac{1}{2}, \phi_{j+\frac{1}{2},k-1}\right) \qquad \left(0, \phi_{j+\frac{1}{2},k-\frac{1}{2}}\right) \qquad \left(1, \phi_{j+\frac{1}{2},k+\frac{1}{2}}\right) \qquad \left(\frac{3}{2}, \phi_{j+\frac{1}{2},k+1}\right).$$

The location of the crossing is similarly obtained by locating a root of the interpolating polynomial.

Although two node-centered and three face-centered level set values are available, they cannot be used to construct a fifth order accurate estimate of the level set crossing since the interpolation of the cell-centered level set to nodes and faces is only fourth order accurate. There are more face-centered, interpolated level set values available than are needed, so the stencils are chosen to be symmetric for simplicity and to avoid possible biasing. Note that using the two averaged node values in constructing the interpolating polynomial guarantees that a polynomial root will be found whenever the sign tests indicate that a face is crossed by the interface. A bracketed secant method was used to compute the interpolating polynomial roots.

The resulting mesh is not quite suitable for use with our curvature discretization. The surface may contain sliver elements, which the curvature discretization is sensitive to, a point which is discussed in more detail in Section 6.3.2. This is handled by eliminating surface elements whose length is less than $0.1\Delta x$. Such elements are merged with a neighboring element by removing an endpoint.

6.3.2 Lagrangian Surface Tension

Let $\mathbf{x}_{k-1}$, $\mathbf{x}_k$, and $\mathbf{x}_{k+1}$ be three consecutive vertices on the Lagrangian surface mesh. These points conceptually lie on a smooth curve $\boldsymbol{\gamma}(s)$, where s is taken to be the arc length parameter for simplicity. The curvature of this curve is then given by $\boldsymbol{\gamma}'' = \kappa\mathbf{n}$. Let $\ell_{k-\frac{1}{2}} = \|\mathbf{x}_k - \mathbf{x}_{k-1}\|$ and $\ell_{k+\frac{1}{2}} = \|\mathbf{x}_{k+1} - \mathbf{x}_k\|$ be the lengths of the edges adjacent to $\mathbf{x}_k$. It will also be convenient to associate some portion of the surface with each of the vertices on the Lagrangian surface mesh. We adopt the convention that a length $\ell_k = \frac{\ell_{k-\frac{1}{2}} + \ell_{k+\frac{1}{2}}}{2}$ of the surface mesh is associated with point $\mathbf{x}_k$. With these definitions, the curvature can be discretized using

$$(\kappa\hat{\mathbf{n}})_k = (\boldsymbol{\gamma}'')_k = \frac{\frac{\mathbf{x}_{k+1}-\mathbf{x}_k}{\ell_{k+\frac{1}{2}}} - \frac{\mathbf{x}_k-\mathbf{x}_{k-1}}{\ell_{k-\frac{1}{2}}}}{\ell_k}. \tag{6.34}$$

This quantity is also known as the surface Laplacian. Since surface tension results in a pressure jump $[p] = \sigma\kappa$ across the interface, the force due to surface tension will be the pressure times the surface area, or

$$\hat{\mathbf{f}}_k = \sigma\ell_k(\kappa\hat{\mathbf{n}})_k = \sigma\left(\frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{\ell_{k+\frac{1}{2}}} - \frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\ell_{k-\frac{1}{2}}}\right). \tag{6.35}$$

Let $\hat{\mathbf{f}} = \mathbf{D}\mathbf{x}$, where $\hat{\mathbf{f}}$ and $\mathbf{x}$ are the vectors of forces and positions and $\mathbf{D}$ is the matrix given by

$$\mathbf{D}_{k,k} = -\sigma\left(\ell^{-1}_{k+\frac{1}{2}} + \ell^{-1}_{k+\frac{1}{2}}\right)\mathbf{I} \qquad \mathbf{D}_{k,k+1} = \mathbf{D}_{k+1,k} = \sigma\ell^{-1}_{k+\frac{1}{2}}\mathbf{I}$$

with $\mathbf{D}_{i,j} = 0$ otherwise. Note that $\mathbf{D}$ depends on $\mathbf{x}$ through edge lengths. The matrix $\mathbf{D}$ is a symmetric, and one can see that $\mathbf{D}$ is in fact negative semidefinite by noting that it can be written as the sum of force contributions from individual segments,

$$\begin{pmatrix} \hat{\mathbf{f}}_k \\ \hat{\mathbf{f}}_{k+1} \end{pmatrix} = -\sigma\ell^{-1}_{k+\frac{1}{2}} \begin{pmatrix} \mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{x}_k \\ \mathbf{x}_{k+1} \end{pmatrix}.$$

These matrices are symmetric negative semidefinite, so their sum $\mathbf{D}$ must also be.

This per-element form of the force is particularly convenient. In the remainder of this section, we focus on a single surface segment and drop indices where no confusion will result. Let ϕ be the potential energy due to surface tension for the element. Recalling that $\ell = \|\mathbf{x}_{k+1} - \mathbf{x}_k\|$, one may define a tangential direction for the element $\mathbf{t} = \ell^{-1}(\mathbf{x}_{k+1} - \mathbf{x}_k)$. The forces due to this segment are related to the potential by $\hat{\mathbf{f}}_k = -\frac{\partial\phi}{\partial\mathbf{x}_k}$ and $\hat{\mathbf{f}}_{k+1} = -\frac{\partial\phi}{\partial\mathbf{x}_k}$. Finally, the potential energy and forces can be written and computed very concisely as

$$\phi = \sigma\ell \qquad \hat{\mathbf{f}}_{k+1} = -\hat{\mathbf{f}}_k = -\sigma\mathbf{t} \tag{6.36}$$

That is, the potential energy due to surface tension is just a constant times the surface area. As one would expect, this potential energy is bounded from below. It is interesting to note that the 2D surface tension force has a form very similar to that of a zero-rest-length spring, whose potential energy is proportional to its length

squared. The force derivatives are given by

$$\frac{\partial \hat{\mathbf{f}}_k}{\partial \mathbf{x}_k} = \frac{\partial \hat{\mathbf{f}}_{k+1}}{\partial \mathbf{x}_{k+1}} = -\frac{\partial \hat{\mathbf{f}}_{k+1}}{\partial \mathbf{x}_k} = -\frac{\partial \hat{\mathbf{f}}_k}{\partial \mathbf{x}_{k+1}} = -\frac{\sigma}{\ell}(\mathbf{I} - \mathbf{t}\mathbf{t}^T) = -\frac{\sigma}{\ell}\mathbf{n}\mathbf{n}^T,$$

where $\mathbf{n}$ is the normal vector defined to be orthogonal to $\mathbf{t}$. This resulting system is symmetric and negative definite. The force derivatives project out and tangential velocity component before computing damping, and the resulting force acts only in the normal direction. Interestingly, a frozen-coefficient treatment of edge lengths treats $\mathbf{D}$ as constant. This leads to $\mathbf{D}$ as an approximation to the force derivatives, which will damp motion in the tangential direction and apply forces with both normal and tangential components. Though not the correct force derivatives, this extra damping may actually be desirable in practice. Our scheme for handling surface tension uses pressures (see Section 6.3.5 for more details on why this is so) and thus effectively discards tangential velocity. This prevents the Lagrangian surface from experiencing tangential velocity, so tangential damping is unnecessary. As such, the correct force derivatives are used in our discretization.

To summarize, a fully implicit treatment and a fully explicit treatment would both compute $\hat{\mathbf{f}}_e$ from (6.10) as

$$(\hat{\mathbf{f}}_e)_k = \sigma \mathbf{t}_{k+\frac{1}{2}} - \sigma \mathbf{t}_{k-\frac{1}{2}} \qquad \ell_{k+\frac{1}{2}} = \|\mathbf{x}_{k+1} - \mathbf{x}_k\| \qquad \mathbf{t}_{k+\frac{1}{2}} = \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{\ell_{k+\frac{1}{2}}}.$$

A fully implicit treatment would compute $\mathbf{D}$ from (6.11) as

$$\mathbf{D}_{k,k} = -\frac{\Delta t \sigma}{\ell_{k-\frac{1}{2}}}\mathbf{n}_{k-\frac{1}{2}}\mathbf{n}_{k-\frac{1}{2}}^T - \frac{\Delta t \sigma}{\ell_{k+\frac{1}{2}}}\mathbf{n}_{k+\frac{1}{2}}\mathbf{n}_{k+\frac{1}{2}}^T \qquad \mathbf{D}_{k,k+1} = \mathbf{D}_{k+1,k} = \frac{\Delta t \sigma}{\ell_{k+\frac{1}{2}}}\mathbf{n}_{k+\frac{1}{2}}\mathbf{n}_{k+\frac{1}{2}}^T$$

where

$$\mathbf{n}_{k+\frac{1}{2}} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \mathbf{t}_{k+\frac{1}{2}}.$$

A fully implicit treatment would compute $\mathbf{C}$ from Section 6.2.6 as

$$\mathbf{C}_{k,k} = -\mathbf{C}_{k,k+1} = \Delta t \sqrt{\frac{\sigma}{\ell_{k+\frac{1}{2}}}} \mathbf{n}_{k+\frac{1}{2}}^T$$

A fully explicit treatment would compute $\mathbf{D} = \mathbf{0}$ and $\mathbf{C} = \mathbf{0}$, making it less stable.

Discretization Accuracy

Consider again the three points on the smooth curve $\boldsymbol{\gamma}(s)$, where s is taken to be the arc length parameter. Then, $\|\boldsymbol{\gamma}'\| = 1$ and $\kappa = |\boldsymbol{\gamma}''|$. Assume that $\mathbf{x}_{k-1} = \boldsymbol{\gamma}(-\delta)$, $\mathbf{x}_k = \boldsymbol{\gamma}(0)$, and $\mathbf{x}_{k+1} = \boldsymbol{\gamma}(\beta\delta)$. Here, β is fixed and the refinement $\delta \to 0$ is considered. This corresponds roughly to choosing edge lengths which differ by a factor of β. The first few terms of the Taylor series expansion of the discretization in (6.34) are

$$\begin{aligned}(\kappa\hat{\mathbf{n}})_k \approx\ & \kappa\mathbf{n} + \frac{\beta-1}{12}(4\dot{\kappa}\mathbf{n} + 3\kappa^2\mathbf{t})\delta + \frac{\beta^2-\beta+1}{12}(2\dot{\kappa}\kappa\mathbf{t} - \ddot{\kappa}\mathbf{n})\delta^2 \\ &+ \frac{\beta-1}{576}\big(3(\beta-1)^2\kappa^4 + 8(\beta^2+1)(3\kappa\ddot{\kappa} + 2\dot{\kappa}^2)\big)\mathbf{t}\delta^3 \\ &- \frac{\beta-1}{360}\big((4\beta^2-5\beta+4)\dot{\kappa}\kappa + 6(\beta^2+1)\dddot{\kappa}\big)\mathbf{n}\delta^3,\end{aligned}$$

where that $\mathbf{n}$ and $\mathbf{t}$ are the normal and tangential directions. There are a few useful observations to make from this Taylor series. The discretization is first order accurate. When $\beta = 1$, so that the surface mesh is uniformly spaced, the first and third order terms vanish, and the discretization is second order accurate. In particular, the error coefficient will be reduced if the discretization is approximately uniform.

The special case of a circle is quite interesting. In that case, κ is constant, so that $\dot{\kappa} = \ddot{\kappa} = \dddot{\kappa} = 0$. The series simplifies significantly to

$$(\kappa\hat{\mathbf{n}})_k \approx \kappa\mathbf{n} + \frac{\beta-1}{4}\kappa^2\delta\mathbf{t} + \frac{(\beta-1)^3}{192}\kappa^4\delta^3\mathbf{t}.$$

If the discretization is uniform, these error terms vanish. What is more, the errors

are purely in the tangential direction. Since our discretization of $\mathbf{H}$ does not transmit (discretized) tangential impulses, the resulting tangential force errors do not feed significantly into the rest of the simulation. This additional accuracy in the case of a circle is observed numerically, and our curvature estimates for a circle correct to roundoff error for a completely uniform discretization whose vertices lie exactly on the circle.

Next the sensitivity of the proposed discretization to errors in particle locations is considered. Errors made along the surface mesh correspond approximately to choosing a different β. Since no attempt is made to keep $\beta = 1$, such errors are not a significant problem. This leaves only the sensitivity of the curvature computation to errors in the normal direction. Consider that three points are chosen on a circle, but the middle point is displaced slightly in the normal direction. That is

$$\mathbf{x}_{k-1} = \begin{pmatrix} r\cos\theta \\ -r\sin\theta \end{pmatrix} \qquad \mathbf{x}_k = \begin{pmatrix} r(1+\epsilon) \\ 0 \end{pmatrix} \qquad \mathbf{x}_{k+1} = \begin{pmatrix} r\cos\phi \\ r\sin\phi \end{pmatrix}.$$

As before, it is assume that $\phi = \beta\theta$, where β is held fixed and θ and ϵ are both very small. Then

$$\begin{aligned}(\kappa\hat{\mathbf{n}})_k \approx \left(\frac{1}{r} + \frac{\epsilon}{\beta r}\left(\frac{2}{\theta^2} + \frac{\beta^2 - 12\beta + 1}{12} + \frac{(7\beta^4 + 10\beta^2 + 7)\theta^2}{2880}\right)\right)\mathbf{n} \\ + \frac{(1-\beta)(1-\epsilon)\theta}{192r}\big(48 + (1-\beta)^2\theta^2\big)\mathbf{t}\end{aligned}$$

Of particular importance is the term $\frac{2\epsilon}{\beta r\theta^2}$. Convergence is only possible if this term vanishes under refinement. Since the surface is a circle of radius r, the angle θ is related to the Eulerian grid spacing Δx by $r\theta \approx \Delta x$. Then, the error term takes the form $\frac{2\epsilon r}{\beta \Delta x^2}$. Since we require the error to be of order $O(\Delta x)$ or better for convergence, it is necessary for $\epsilon = O(\beta \Delta x^3)$. This is the level of sensitivity to Δx that one would expect from a second derivative. The tolerable error is also sensitive to the irregularity β of the discretization, and very small elements will result in poor accuracy. For this reason, it is important to avoid sliver elements in our boundary mesh.

Our discretization of curvature can be obtained from the potential energy in (6.36). The Taylor series expansion of the potential energy discretization along the portion of the curve between $\boldsymbol{\gamma}(0)$ and $\boldsymbol{\gamma}(s)$ is

$$\phi \approx \sigma s - \frac{1}{24}\sigma\kappa^2 s^3 - \frac{1}{24}\sigma\kappa\dot{\kappa}s^4.$$

Since $\boldsymbol{\gamma}(s)$ is assumed to be parameterized in terms of its arc length, the exact length of this portion of the curve is s. Thus, the estimate of surface area is second order accurate, which leads to a first order curvature force. This suggests that if a more accurate curvature force computation is desired, a third or higher order approximation of the surface area should be sought.

6.3.3 Eulerian Degrees of Freedom

We require additional accuracy at the fluid surface to obtain accurate surface tension. There are a vast array of existing discretization schemes designed for irregular Dirichlet boundary conditions that are able to achieve higher accuracy. For example, [98] utilized a discontinuous-Galerkin scheme near the boundary to increase the available degrees of freedom, avoid locking, and achieve second order convergence. Similarly, there are second order discretizations available on regular grids, such as the finite differencing dsicretization of [57]. To meet the needs of our Lagrangian-Eulerian coupling, we have opted to use a first order Dirichlet variant of the second order finite volume Neumann discretization [107].

Our Eulerian discretization is based on a standard MAC grid discretization, but it is altered slightly near the interface. We refer to any cell through which the interface passes as a cut cell, and we refer to any face through which the interface passes as a cut face. We refer to cells/faces entirely inside/outside the fluid as interior/exterior cells/faces. In addition to this, additional fluid degrees of freedom are defined that lie approximately on the interface, and we will refer to these as coupling faces. These degrees of freedom are illustrated in Figure 6.2.

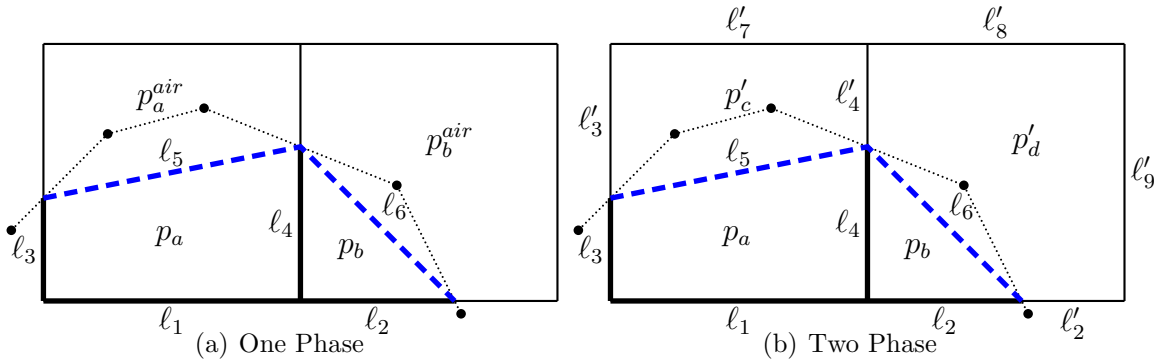


Figure 6.2: Two cells cut by a Lagrangian surface mesh. A face is constructed in each cut cell (shown as a dashed line) connecting the point where the surface mesh enters the cell and the point where it leaves. These two faces are referred to as coupling faces and have lengths ℓ_5 and ℓ_6. There are three cut faces with lengths ℓ_2, ℓ_3, and ℓ_4. Finally, there is one interior face with length $\ell_1 = \Delta x$. Each cut cell has a pressure, which in this case are labeled p_a and p_b. In the one-phase case, there are Dirichlet pressure boundary conditions p_a^{air} and p_b^{air} outside the coupling faces. In the two-phase case, there are additional pressure degrees of freedom in the outside portion of each cut cell instead, labelled p_c and p_d, and there are additionally three exterior faces with lengths $\ell_7' = \ell_8' = \ell_9' = \Delta x$.

One Phase In the one-phase case, each interior cell and each cut cell contains a pressure degree of freedom, and each interior face (Figure 6.2, ℓ_1), cut face (Figure 6.2, ℓ_2, ℓ_3, and ℓ_4), and coupling face (Figure 6.2, ℓ_5 and ℓ_6) contains one velocity degree of freedom. In the case of coupling faces, the velocity degree of freedom is taken to be directed in the outward normal direction. Outside the coupling faces, there is a Dirichlet pressure.

Two Phase In the two-phase case, each cut cell contains an inside pressure and an outside pressure. All other cells contain one pressure degree of freedom. Every grid face and every coupling face contains one velocity degree of freedom. Note that one may allow the velocity field to support a discontinuity in tangential velocity by simply providing each cut face with an inside velocity and an outside velocity in much the same way as is done for pressure. Since surface tension will be applied to the coupling faces, there will be a pressure jump across this face. Thus, one cannot assume a continuous pressure field, and cut cells must contain separate inside and

outside pressures. As in the one-phase case, the velocity degree of freedom for a coupling face is taken to be directed in the outward normal direction.

6.3.4 Gradient and Fluid Mass

The volume weighted gradient $\mathbf{G}$ and the fluid mass β are required for our discretization. First consider the two-phase case. The mass for a interior or exterior faces is just ρV, where V is the cell volume and ρ is the density at that face. For a cut face with inside length and density ℓ and ρ and outside length and density ℓ' and ρ' the mass is $\beta = (\rho\ell + \rho'\ell')\Delta x$. For a coupling face with length ℓ, the mass will be $\beta = \frac{1}{2}\ell\Delta x(\rho + \rho')$. The one-phase case is obtained by letting the exterior be massless, so that $\rho' = 0$. Intuitively, each face has a dual cell associated with it. Some portion of that dual cell may be in the inside region, and some portion may be in the outside region. The masses above correspond to dividing the dual cell up into inside and outside portions, simplifying things by making all the pieces rectangular. This discretization of mass is only first order accurate, since it double-counts mass on the boundary and treats the boundary as linear rather than computing the mass based on the curvature. A higher order discretization of mass will require a more detailed and careful treatment.

The weights on the volume-weighted gradient stencil are just the lengths of the faces. The stencil entry G_{fc} is positive if the velocity u_f represents flow into cell c and negative otherwise. One can think of this as corresponding to a constant pressure in each (possibly cut) cell. Alternatively, one can think of this instead in terms of the divergence operator, where the faces are now used to compute the flux into the (possibly cut) cell.

As an example in the one-phase case, consider the discretization in Figure 6.2(a), where faces and cells are labeled with the index of the length or pressure at that location in the figure. Then, $\beta_{55} = \frac{1}{2}\ell_5\rho_5\Delta x$, $\beta_{66} = \frac{1}{2}\ell_6\rho_6\Delta x$, and $\beta_{kk} = \ell_k\rho_k\Delta x$ for $1 \leq k \leq 4$. The volume-weighted gradient stencil entries are $G_{1a} = \ell_1 = \Delta x$,

$G_{3a} = \ell_3$, $G_{4a} = -\ell_4$, $G_{5a} = -\ell_5$, $G_{2b} = \ell_2$, $G_{4b} = \ell_4$, and $G_{6b} = -\ell_6$.

As a two-phase example, consider the discretization in Figure 6.2(b). Here, primes indicate quantities corresponding to the outside phase and subscripts indicate velocity or pressure degrees of freedom. Then, $\beta_{55} = \frac{1}{2}\ell_5(\rho_5 + \rho'_5)\Delta x$, $\beta_{66} = \frac{1}{2}\ell_6(\rho_6 + \rho'_6)\Delta x$, $\beta_{11} = \ell_1\rho_1\Delta x$, $\beta_{kk} = (\ell_k\rho_k + \ell'_k\rho'_k)\Delta x$ for $2 \leq k \leq 4$, and $\beta_{kk} = \ell'_k\rho'_k\Delta x$ for $7 \leq k \leq 9$. The gradient stencil entries are $G_{1a} = \ell_1 = \Delta x$, $G_{3a} = \ell_3$, $G_{4a} = -\ell_4$, $G_{5a} = -\ell_5$, $G_{2b} = \ell_2$, $G_{4b} = \ell_4$, $G_{6b} = -\ell_6$, $G_{3c} = \ell'_3$, $G_{4c} = -\ell'_4$, $G_{5c} = \ell'_5$, $G_{7c} = -\ell'_7 = -\Delta x$, $G_{2d} = \ell'_2$, $G_{4d} = \ell'_4$, $G_{6d} = \ell'_6$, $G_{8d} = -\ell'_8 = -\Delta x$, and $G_{9d} = -\ell'_9 = -\Delta x$.

6.3.5 Eulerian-Lagrangian Interpolation

The role of the velocity interpolation operator $\mathbf{H}$ is to interpolate Eulerian velocity degrees of freedom to Lagrangian degrees of freedom, noting that $\mathbf{H}^T$ serves the complementary role of distributing force from the Lagrangian degrees of freedom back to the Eulerian degrees of freedom. In principle, any interpolation operator could be used for this purpose, such as one constructed from delta functions. In practice, using such a discretization with surface tension produces large parasitic currents, and we are lead to formulate $\mathbf{H}$ with the special needs of surface tension in mind. If there are multiple forces on a Lagrangian mesh, each force can utilize its own discretization of $\mathbf{H}$, so formulating $\mathbf{H}$ specifically for one force is not problematic.

Formulating a discretization of surface tension that eliminates parasitic currents entirely is quite difficult. This task is well beyond the scope of interest, as acceptable results can be obtained even with parasitic currents, provided their magnitude is kept at a manageable level. There are many sources for parasitic currents. One source of parasitic currents is simply an inaccurate curvature computation [54]. While our curvature computation is in general only first order accurate, we have found it to produce acceptable results.

A more serious source of parasitic currents is related to the ability of the pressure to discretely balance out surface tension, and this is the issue we seek to address.

Consider the case of a static circle, where a force $\mathbf{H}^T\hat{\mathbf{f}}$ is produced by surface tension, and a force $-\mathbf{G}\hat{\mathbf{p}}$ is produced by the resulting pressure field to resist it. The net force on the fluid is $\mathbf{H}^T\hat{\mathbf{f}} - \mathbf{G}\hat{\mathbf{p}}$. Since the pressure projection step can be formulated as a kinetic energy minimization problem [15], if the fluid is at rest and a suitable pressure field $\hat{\mathbf{p}}$ exists such that $\mathbf{H}^T\hat{\mathbf{f}} = \mathbf{G}\hat{\mathbf{p}}$, then the result of the pressure projection will be to choose such a pressure. The result would be to produce a zero velocity field. Such a $\hat{\mathbf{p}}$ exists precisely when $\mathbf{H}^T\hat{\mathbf{f}}$ lies in the column space of $\mathbf{G}$. This suggests a formulation of a form similar to $\mathbf{H}^T = \mathbf{G}\hat{\mathbf{H}}^T$, for some operator $\hat{\mathbf{H}}$, and this is the approach that was taken. Note that $\mathbf{H}^T = \mathbf{G}\hat{\mathbf{H}}^T$ itself is not quite what is desired, since then pressure would always cancel surface tension, which is not the case. The force distribution operator $\mathbf{H}^T$ is formulated directly, leaving the velocity interpolation operator to be defined by its transpose.

Since the Lagrangian force will be applied to the Eulerian degrees of freedom using the volume-weighted gradient operator $\mathbf{G}$, the Lagrangian force must be translated to a pseudo-pressure via $\hat{\mathbf{H}}^T$. In fact, it is more straightforward to construct the gradient times pseudo-pressure directly on coupling faces. Our first step, then, is to define a force density everywhere on the surface. In Section 6.3.2, the particle $\mathbf{x}_k$ is associated with a length of the surface equal to half its neighboring segments, so that the force density at the surface can be defined by

$$\hat{\mathbf{f}}_k^\rho = \frac{\hat{\mathbf{f}}_k}{\ell_k} = \sigma(\kappa\hat{\mathbf{n}})_k, \tag{6.37}$$

where as before

$$\ell_k = \frac{\ell_{k-\frac{1}{2}} + \ell_{k+\frac{1}{2}}}{2} = \frac{\|\mathbf{x}_{k+1} - \mathbf{x}_k\| + \|\mathbf{x}_k - \mathbf{x}_{k-1}\|}{2}.$$

The force $\hat{\mathbf{f}}_k$ was defined in (6.35), and $(\kappa\hat{\mathbf{n}})_k$ was defined in (6.34). Note that $\hat{\mathbf{f}}_k^\rho$ is a vectorial quantity rather than a pseudo-pressure. This force density $\hat{\mathbf{f}}^\rho$ is extended from the Lagrangian degrees of freedom to the entire surface mesh by linear interpolation.

The next step requires the per-cell coupling faces illustrated in Figure 6.2 as ℓ_5 and ℓ_6. Let $\mathbf{n}_i$ be the area-weighted surface normal for the coupling face in cell i. For cell i, let S_i be the list of segments k of the Lagrangian mesh (with endpoints $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$) that intersect cell i, and let $\alpha_{k,i}$ and $\beta_{k,i}$, where $0 \leq \alpha_{k,i} \leq \beta_{k,i} \leq 1$, be the barycentric coordinates on the segment k indicating the portion of the segment contained within the cell i. That is, if $\alpha_{k,i} = 0$, then $\mathbf{x}_k$ is inside cell i. Otherwise, $\mathbf{x}_k + \alpha_{k,i}(\mathbf{x}_{k+1} - \mathbf{x}_k)$ is on the boundary of cell i. Similarly, if $\beta_{k,i} = 1$, then $\mathbf{x}_k$ is inside cell i. Otherwise, $\mathbf{x}_{k+1} + \beta_{k,i}(\mathbf{x}_k - \mathbf{x}_{k+1})$ is on the boundary of cell i. With these barycentric weights, the coupling face normal $\mathbf{n}_i$ and its unit normal are easily computed as the sum of a subset of the individual area weighted normals from the Lagrangian mesh as

$$\mathbf{n}_i = \sum_{k \in S_i} (\beta_{k,i} - \alpha_{k,i}) \mathbf{R}(\mathbf{x}_{k+1} - \mathbf{x}_k) \qquad \mathbf{R} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \qquad \hat{\mathbf{n}}_i = \frac{\mathbf{n}_i}{\|\mathbf{n}_i\|}.$$

Next, a force $\hat{\mathbf{f}}_i$ is computed for the cell by integrating the surface force density over the portion of the surface in the cell i

$$\hat{\mathbf{f}}_i = \sum_{k \in S_i} \int_{\alpha_{k,i}}^{\beta_{k,i}} \hat{\mathbf{f}}_\rho \, dA.$$

The volume-weighted gradient of pseudo-pressure $G_{kc} p_i$ in cell i would then be

$$G_{kc} p_i = \hat{\mathbf{n}}_i \cdot \hat{\mathbf{f}}_i.$$

Here, the dot product with $\hat{\mathbf{n}}_i$ removes the tangential components of the force.

For convenience, the explicit form of $(\mathbf{H}^T)_{i,k}$, the operator such that $(\mathbf{H}^T)_{i,k} \hat{\mathbf{f}}_k$ is the force at the coupling face in cell i from particle k, is provided as

$$\mathbf{H}_{k,i} = \left(\frac{\beta_{k-1,i} + \alpha_{k-1,i}}{2} \right) \frac{(\beta_{k-1,i} - \alpha_{k-1,i}) \ell_k}{\ell_{k-\frac{1}{2}}} \hat{\mathbf{n}}_i + \left(1 - \frac{\beta_{k,i} + \alpha_{k,i}}{2} \right) \frac{(\beta_{k,i} - \alpha_{k,i}) \ell_k}{\ell_{k+\frac{1}{2}}} \hat{\mathbf{n}}_i, \tag{6.38}$$

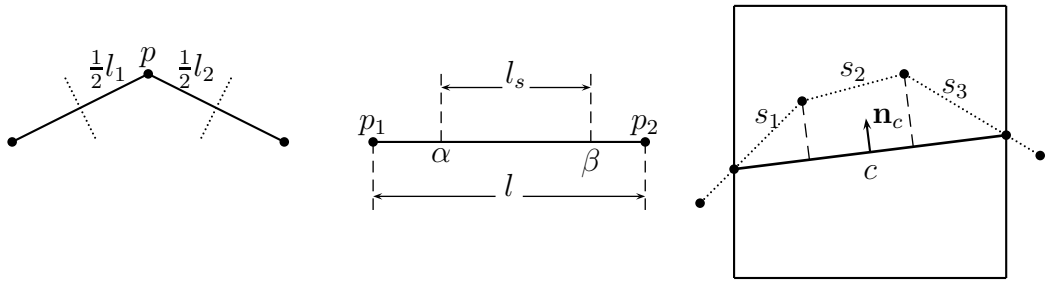
where the convention that $\alpha_{k,i} = \beta_{k,i} = 0$ for $k \notin S_i$ has been used. The first term in

(6.38) is the contribution from the edge between $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$, and the second term is the contribution from the edge between $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$. This formula can be interpreted more intuitively. The first term corresponds to a segment that touches the cell i over some fraction of its length. The vector $\hat{\mathbf{n}}_i$ is the direction over which the force is applied, and it is determined by the orientation of the Eulerian coupling face. The average $\frac{\beta_{k-1,i}+\alpha_{k-1,i}}{2}$ is the location of the middle of this partial segment. If the portion of the segment in cell i is near the node k, then $\beta_{k-1,i}$ and $\alpha_{k-1,i}$ will be larger. If the portion in cell i is closer to the other endpoint, then the average will be smaller. This distributes more of the pressure force to the segment onto the endpoint of the segment that is closer. The factor $\beta_{k-1,i} - \alpha_{k-1,i}$ scales down the area over which the pressure is applied, and thus the magnitude of the final force, based on the fraction of the segment over which the pressure from cell i applies. The remaining term is the ratio $\frac{\ell_k}{\ell_{k-\frac{1}{2}}}$, which is effectively a correction for when the segments adjacent to particle k are not the same length.

Note that $\hat{\mathbf{n}}_i \cdot \hat{\mathbf{f}}_i$ guarantees that only the normal component of the force from the Lagrangian mesh will be transfered back to the fluid, a property entirely consistent with a description of surface tension as a pressure jump. Substituting $\mathbf{H}^T = \mathbf{G}\hat{\mathbf{H}}^T$ into (6.9) produces

$$\mathbf{u}^{n+1} = \mathbf{u}^\star - \boldsymbol{\mathcal{B}}^{-1}\mathbf{G}(\hat{\mathbf{p}} - \Delta t\hat{\mathbf{H}}\hat{\mathbf{f}}_e - \Delta t\hat{\mathbf{H}}\mathbf{D}\mathbf{H}\mathbf{u}^{n+1}) - \boldsymbol{\mathcal{B}}^{-1}\mathbf{G}_\mu^T\mathbf{G}_\mu\mathbf{u}^{n+1}.$$

Written in this way, the surface tension force takes the form of a jump in pressure near the interface. Since $\mathbf{H}^T$ does not transfer tangential forces, the interpolation operator $\mathbf{H}$ will not transfer tangential velocities. This property is also consistent with surface tension, which is sensitive to the interface location but not motion tangential to the interface.



(a) Distribution (b) Integration (c) Summation and projection

Figure 6.3: Distribution of forces from the Lagrangian mesh to the Eulerian mesh. A force defined on particle p is first distributed onto half length of its neighboring segments to define a force density at p. Then the force density is be integrated over the portion of a segment between barycentric weights α and β which indicate the portion of the segment inside the grid cell under consideration. Finally, the integrated force on segments s_1, s_2 and s_3 within a cut cell will be added and projected onto the coupling face c given its unit normal $\mathbf{n}_c$.

6.3.6 Note on Second Order Neumann Poisson Discretization

Our discretization of gradient and mass in the one-phase case is very closely related to the Neumann Poisson discretization of [107]. Let ℓ_f be the length of face f and $s_{fc} = 1$ if the velocity u_f represents flow into cell c, $s_{fc} = -1$ if the velocity u_f represents flow out of cell c, and $s_{fc} = 0$ otherwise. Let $r_f = \frac{1}{2}$ if f is a coupling face and $r_f = 1$ otherwise. With these, our gradient is $G_{fc} = \ell_f s_{fc}$, and our mass is $\beta_{ff} = r_f \ell_f \rho_f \Delta x$. In the absence of surface tension or viscosity, the system that must be solved and the subsequent pressure application are

$$-\mathbf{G}^T \boldsymbol{\beta}^{-1} \mathbf{G} \hat{\mathbf{p}} = -\mathbf{G}^T \mathbf{u}^\star \qquad \mathbf{u}^{n+1} = \mathbf{u}^\star - \boldsymbol{\beta}^{-1} \mathbf{G} \hat{\mathbf{p}}$$

In terms of components, this is

$$-\sum_{f,c} G_{fc'} \beta_{ff}^{-1} G_{fc} \hat{p}_c = -\sum_f G_{fc'} u_f^\star \qquad u_f^{n+1} = u_f^\star - \sum_c \beta_{ff}^{-1} G_{fc} \hat{p}_c.$$

Substituting in the discretization produces

$$-\sum_{f,c} s_{fc'} s_{fc} \frac{\ell_f}{\rho_f r_f} \frac{\hat{p}_c}{\Delta x} = -\sum_f s_{fc'} \ell_f u_f^\star \qquad u_f^{n+1} = u_f^\star - \sum_c \frac{s_{fc}}{\rho_f r_f} \frac{\hat{p}_c}{\Delta x}.$$

Our discretization corresponds to a Dirichlet boundary condition. The boundary condition can be converted into a Neumann condition by prescribing fixed values for the coupling faces, so that those values move to the right hand side. Assume for simplicity that these velocities are zero. Since only cut and interior faces remain, $r_f = 1$, and the discretization simplifies to

$$-\sum_{f,c} s_{fc'} \frac{\ell_f}{\rho_f} \frac{\sum_c s_{fc} \hat{p}_c}{\Delta x} = -\sum_f s_{fc'} \ell_f u_f^\star \qquad u_f^{n+1} = u_f^\star - \sum_c \frac{s_{fc}}{\rho_f} \frac{\hat{p}_c}{\Delta x}.$$

The Poisson discretization is the same as equation (2) in [107]. The pressure application equation is the same as that used by [107]. In some sense, our Dirichlet pressure discretization is compatible with the Neumann pressure discretization in [107]. This also demonstrates that the Neumann discretization can be written in the form of (6.3.6). This form is particularly useful, since it allows the discretization of [107] to be used with the FSI formulations of [124]. Given our rather simple choice of the mass at coupling faces, our discretization will only be first order accurate as mentioned in Section 6.3.4. We leave it for future work to try to extend our Dirichlet discretization to second order.

6.3.7 Note on Second Order Dirichlet Poisson Discretization

The second order Dirichlet Poisson discretization of [57] is used for the sake of comparison. As with with the proposed scheme, it is implemented within the framework of [124], which requires the discretization to factor as in (6.3.6). It also requires the application of nonzero Dirichlet boundary conditions, a task not addressed in [124]. The omission is straightforward to address by adding $-\boldsymbol{\beta}^{-1}\mathbf{G}_{bc}\hat{\mathbf{p}}_{bc}$ to $\mathbf{u}^\star$. Here $\mathbf{G}_{bc}$ is a continuation of the gradient stencil to include pressures other than the degrees

of freedom, and $\hat{\mathbf{p}}_{bc}$ are the corresponding Dirichlet pressures. One may obtain this result by replacing the gradient and pressure with extended versions that include both degrees of freedom and ghost pressures. Then, the ghost pressure terms are moved to the right hand side, where it is found that they can be folded into $\mathbf{u}^\star$ as described. It is important to note that this correction to $\mathbf{u}^\star$ occurs on the system right hand side as well as in the pressure update equation. The next issue to address is that the pressure jump conditions are not imposed at cells but rather at the interface location. This is addressed by simply multiplying the appropriate pressure jump by each entry in $\mathbf{G}_{bc}$ rather than using the same pressure jump for all faces adjacent to the Dirichlet cell as would occur if ghost pressure jumps were computed, followed by a matrix vector multiply.

The final issue to address is the factorization of the second order Dirichlet Poisson operator. The appropriate discretizations are $G_{fc} = s_{fc}\Delta x$ and $\beta_{ff} = \ell_f \rho_f \Delta x$, where $\ell = \theta \Delta x$. Indeed, the Poisson operator $\mathbf{G}^T \boldsymbol{\beta}^{-1} \mathbf{G}$ simplifies to $\sum_f \frac{s_{fc'} s_{fc}}{\theta_f \rho_f}$, which is just equation (21) in [57], except that our discretization is volume weighted and thus scaled by Δx^2.

Implementing this discretization in the context of [124] has an interesting additional benefit. This framework allows us to couple the pressure and viscosity systems, something that could not be done by [86].

6.4 Examples

6.4.1 Interface and Curvature Accuracy

The purpose of this example is to demonstrate the accuracy of our method for computing the interface location and for computing the curvature. We use a dimensionless domain $[0, 8] \times [-2, 2]$ with resolution $2N \times N$. The fluid region is taken to be the portion of the domain such that $y \geq sin(x)$. We only consider the portion of the fluid interface that is at least $8\Delta x$ from the domain boundary to avoid boundary artifacts.

Third Order Interface								
N	$L^\infty(\epsilon_\Gamma)$	order	$L^1(\epsilon_\Gamma)$	order	$L^\infty(\epsilon_\kappa)$	order	$L^1(\epsilon_\kappa)$	order
40	$1.299 \cdot 10^{-5}$	-	$3.214 \cdot 10^{-6}$	-	$3.375 \cdot 10^{-2}$	-	$9.626 \cdot 10^{-3}$	-
80	$1.478 \cdot 10^{-6}$	3.14	$2.838 \cdot 10^{-7}$	3.50	$2.063 \cdot 10^{-2}$	0.71	$4.931 \cdot 10^{-3}$	0.97
160	$1.783 \cdot 10^{-7}$	3.05	$2.637 \cdot 10^{-8}$	3.43	$1.065 \cdot 10^{-2}$	0.95	$2.350 \cdot 10^{-3}$	1.07
320	$2.230 \cdot 10^{-8}$	3.00	$2.827 \cdot 10^{-9}$	3.22	$5.258 \cdot 10^{-3}$	1.02	$1.127 \cdot 10^{-3}$	1.06
Fourth Order Interface								
N	$L^\infty(\epsilon_\Gamma)$	order	$L^1(\epsilon_\Gamma)$	order	$L^\infty(\epsilon_\kappa)$	order	$L^1(\epsilon_\kappa)$	order
40	$1.295 \cdot 10^{-5}$	-	$3.012 \cdot 10^{-6}$	-	$2.876 \cdot 10^{-2}$	-	$8.497 \cdot 10^{-3}$	-
80	$8.227 \cdot 10^{-7}$	3.98	$1.848 \cdot 10^{-7}$	4.03	$1.425 \cdot 10^{-2}$	1.01	$4.318 \cdot 10^{-3}$	0.98
160	$5.179 \cdot 10^{-8}$	3.99	$1.166 \cdot 10^{-8}$	3.99	$7.636 \cdot 10^{-3}$	0.90	$2.043 \cdot 10^{-3}$	1.08
320	$3.431 \cdot 10^{-9}$	3.92	$7.468 \cdot 10^{-10}$	3.96	$3.788 \cdot 10^{-3}$	1.01	$9.842 \cdot 10^{-4}$	1.05

Table 6.1: The interface error ϵ_Γ and curvature error ϵ_κ are shown for third and fourth order interfaces computed from the function $y = \sin x$.

The initial level set is computed computed numerically to be exactly a signed distance function. Then, the interface is computed as in Section 6.3.1. In doing so, a number of points (x, y) are computed for the Lagrangian surface approximation. We compute the error for each of these points as $\epsilon_\Gamma = |y - \sin x|$. Then, we compute the curvature times normal $(\kappa\hat{\mathbf{n}})_k$ from the resulting Lagrangian mesh using (6.34). We compute the error estimate as $\epsilon_\kappa = \left| \|(\kappa\hat{\mathbf{n}})_k\| - |\sin x|(1 + \cos^2 x)^{-\frac{3}{2}} \right|$. The L^∞ and L^1 errors for ϵ_Γ and ϵ_κ are shown in Table 6.1 for both the third and fourth order interface computations.

6.4.2 Stationary Circle

Perhaps the most important analytic test that exists for surface tension is that of a sphere in equilibrium. This example is considered in [77, 83, 145, 86], and we use the parameters of [77]. Our domain is $[0, 1] \times [0, 1]$. Our fluid has the unitless parameters $\rho = 10^4$, $\mu = 1$, $\sigma = 1$. The circle radius is $r = 0.25$. The analytic solution is to have a constant pressure $p = \frac{\sigma}{r}$, where the outside pressure is taken to be zero. The velocity should be zero everywhere, but in practice numerical methods generate nonzero parasitic currents. Both the L^1 and L^∞ norms of the velocity error

	One Phase					
	Pressure Jump		Level Set		Front Tracking	
Δx^{-1}	L^1	L^∞	L^1	L^∞	L^1	L^∞
20	$9.40 \cdot 10^{-5}$	$3.20 \cdot 10^{-4}$	$1.07 \cdot 10^{-4}$	$3.70 \cdot 10^{-4}$	$1.83 \cdot 10^{-7}$	$9.59 \cdot 10^{-7}$
40	$5.66 \cdot 10^{-6}$	$4.73 \cdot 10^{-5}$	$5.39 \cdot 10^{-6}$	$2.64 \cdot 10^{-5}$	$4.41 \cdot 10^{-8}$	$3.28 \cdot 10^{-7}$
80	$4.39 \cdot 10^{-7}$	$6.56 \cdot 10^{-6}$	$2.38 \cdot 10^{-7}$	$2.37 \cdot 10^{-6}$	$4.16 \cdot 10^{-9}$	$3.11 \cdot 10^{-8}$
160	$1.58 \cdot 10^{-7}$	$1.76 \cdot 10^{-6}$	$1.74 \cdot 10^{-8}$	$2.71 \cdot 10^{-7}$	$5.20 \cdot 10^{-10}$	$5.54 \cdot 10^{-9}$
Order	3.1	2.5	4.2	3.5	2.9	2.6

Table 6.2: Parasitic current magnitudes for a one-phase stationary circle.

	Two Phase			
	Level Set		Front Tracking	
Δx^{-1}	L^1	L^∞	L^1	L^∞
20	$8.57 \cdot 10^{-5}$	$2.48 \cdot 10^{-4}$	$2.76 \cdot 10^{-7}$	$8.32 \cdot 10^{-7}$
40	$7.56 \cdot 10^{-6}$	$3.33 \cdot 10^{-5}$	$3.72 \cdot 10^{-8}$	$1.60 \cdot 10^{-7}$
80	$1.15 \cdot 10^{-7}$	$1.31 \cdot 10^{-6}$	$4.90 \cdot 10^{-9}$	$1.54 \cdot 10^{-8}$
160	$1.04 \cdot 10^{-8}$	$1.58 \cdot 10^{-7}$	$1.25 \cdot 10^{-10}$	$1.33 \cdot 10^{-9}$
Order	4.5	3.6	3.6	3.1

Table 6.3: Parasitic current magnitudes for a two-phase stationary circle.

are shown. The L^1 error is defined to be $\frac{1}{N}\sum_f |u_f|$, where f runs over all faces, u_f is the velocity component stored at that face, and $N = \sum_f 1$. The L^∞ error is defined to be $\max_f |u_f|$. In each case, the simulation is run until time $t = 5$ using a time step $\Delta t = 0.2\Delta x$, and all errors are computed at the final time. The results are shown in Table 6.2 and Table 6.3 along with convergence order estimates. Note that the pressure jump scheme that we have chosen for comparison is known to produce parasitic currents as much as three orders of magnitude lower than would be achieved from a delta function formulation [86].

6.4.3 Oscillating Circle

In this example we consider the problem of a deformed circle oscillating under surface tension. This problem has been considered elsewhere [145, 48]. The initial configuration consists of a deformed circle described in polar coordinates by $r = \hat{r}(1+\epsilon\cos m\theta)$,

where $m \geq 2$ is an integer indicating the oscillation mode, $\hat{r}$ is the unperturbed radius, and ϵ is the magnitude of the initial perturbation. The approximate frequency and velocity field in the absence of viscosity (see [94]) are

$$\omega^2 = \frac{\sigma(m^2-1)m}{\rho\hat{r}^3} \qquad \mathbf{u} = -\epsilon\omega\hat{r}\left(\frac{r}{\hat{r}}\right)^{m-1}\sin\omega t\begin{pmatrix}\cos(1-m)\theta\\ \sin(1-m)\theta\end{pmatrix}.$$

We note that this is only an approximate solution, and its accuracy improves as $\epsilon \to 0$. Following [48], we use a dimensionless $[0,1]\times[0,1]$ domain with dimensionless parameters $\hat{r} = \frac{1}{3}$, $m = 2$, $\rho = 27$, $\sigma = \frac{2}{3}$, and $\mu = 0$. Several choices of ϵ are considered. The (approximate) analytic period with these parameters is π. The first test was run with $\epsilon = 0.05$, where the period of the first oscillation and the x component of the fluid velocity were determined at the location $(0.75, 0.5)$ at time $t = 0.5$ to establish convergence. In the second test, we consider the refinement of ϵ with the resolution fixed to demonstrate convergence to the analytic analytic solution as the solution itself becomes more accurate. Here we determine the period of the first oscillation, which does not depend on ϵ, and the relative error at time $t = 0.5$. The relative error is taken to be the velocity error, computed for each velocity degree of freedom, divided by the (approximate) maximum analytic velocity magnitude $\epsilon\omega\hat{r}|\sin\omega t|$. Note that the solution scales down with ϵ, but the relative velocity does not, so that a decreasing relative velocity error indicates actual convergence.

The results of both tests are shown in Table 6.4. The first simulation is a pressure jump formulation of surface tension applied using a second order accurate Dirichlet boundary condition which we use for comparison purposes. The second scheme is the proposed method with a surface mesh constructed from the level set, and the third scheme is the proposed method with a front-tracked surface mesh. In the refinement test, the velocities in all three cases appear to be converging first order to a value around $-2.063 \cdot 10^{-2}$. Thus, the three methods are all consistent, at least with each other. Moreover, the pressure jump scheme is consistently more accurate. This is not particularly surprising, since it uses a second order curvature and applies it with a second order boundary condition. This test does not indicate convergence to the

Refinement Test						
	Pressure Jump		Level Set		Front Tracking	
N	T	u	T	u	T	u
50	3.188	$-2.074 \cdot 10^{-2}$	3.277	$-1.916 \cdot 10^{-2}$	3.222	$-1.945 \cdot 10^{-2}$
100	3.192	$-2.069 \cdot 10^{-2}$	3.289	$-1.992 \cdot 10^{-2}$	3.224	$-2.002 \cdot 10^{-2}$
200	3.185	$-2.066 \cdot 10^{-2}$	3.171	$-2.027 \cdot 10^{-2}$	3.198	$-2.032 \cdot 10^{-2}$
Epsilon Test						
	Pressure Jump		Level Set		Front Tracking	
ϵ	T	$L^\infty(\mathbf{u})$	T	$L^\infty(\mathbf{u})$	T	$L^\infty(\mathbf{u})$
0.05	3.185	$5.487 \cdot 10^{-3}$	3.171	$1.021 \cdot 10^{-2}$	3.198	$9.712 \cdot 10^{-3}$
0.025	3.162	$2.277 \cdot 10^{-3}$	3.166	$5.718 \cdot 10^{-3}$	3.180	$6.412 \cdot 10^{-3}$
0.0125	3.152	$8.749 \cdot 10^{-4}$	3.203	$4.697 \cdot 10^{-3}$	3.168	$5.532 \cdot 10^{-3}$

Table 6.4: A deformed circle oscillates under surface tension. The maximum deformation as a fraction of the radius is ϵ. In the refinement test, the spatial and temporal resolution are increased while ϵ is held fixed to demonstrate convergence of the numerical methods. In the epsilon test, the deformation is decreased, demonstrating that as the approximate analytic solution becomes more accurate it approaches the numerical results. The analytic period T in all cases is π.

analytic solution, as the analytic solution is only approximate. Note that the periods of all three methods are near the (approximate) analytic value of π, but these values appear to be noisy in all three schemes, and no clear order of convergence can be deduced in any of the schemes. The periods produced for the pressure jump scheme are very close to the values predicted in [48], which uses essentially the same scheme.

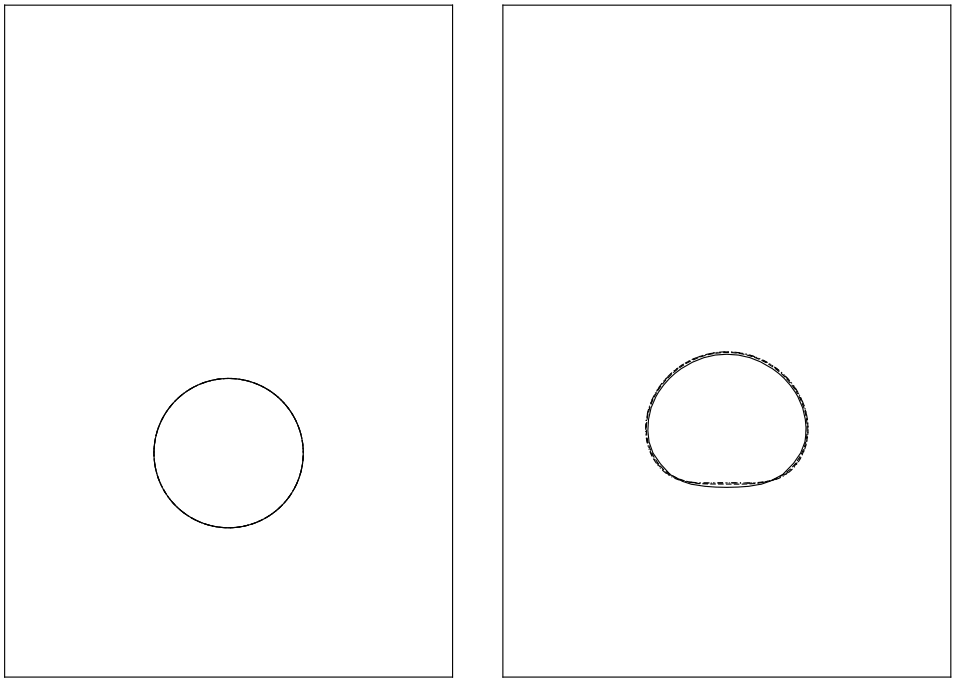
The second test effectively refines the analytic solution to make it more accurate. This allows us to test that the schemes are converging to the correct results. The decreasing velocity errors indicate that the analytic solution and the numerical solutions are approaching each other. The velocity errors are approximately first order for the pressure jump formulation but appear to be less than first order for the proposed method, suggesting that the proposed schemes are not yet fully in the convergence regime. This fact is even more evident from the periods. The periods for the pressure jump are converging first order, but they remain noisy in the proposed schemes. We also observed noisy periods when we tested the pressure jump formulation with a first order accurate discretization of the irregular Dirichlet boundary condition.

6.4.4 Rising Bubble–Stability

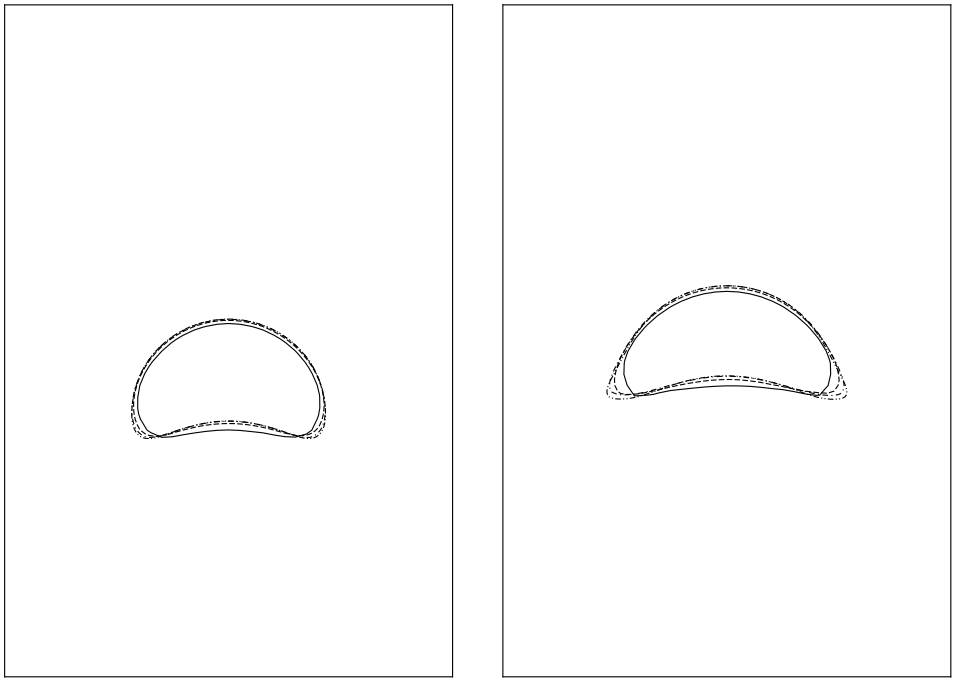
In this example, we test the stability of our method concerning a rising bubble in a heavier fluid, as also considered in [77]. We use the same parameters as in [77]. The domain is $[0\,m, 1\,m] \times [0\,m, 2\,m]$. The bubble is given an initial radius of $r = 0.2\,m$ and is placed at $[0.5\,m, 0.5\,m]$. The density of heavier fluid is $\rho_{water} = 10^4\,kg\,m^{-2}$, and the density of the lighter fluid is $\rho_{air} = 10^3\,kg\,m^{-2}$. The viscosity of both fluids is $\mu = 1\,kg\,s^{-1}$. The surface tension coefficient is $\sigma = 0.5\,kg\,s^{-2}$. We use a gravity of $g = -8 \cdot 10^{-4}\,m\,s^{-2}$. We perform computations on a 80×160 mesh. As discussed in [77], the capillary time step restriction is $\Delta t = 0.056\,s$, and they achieved stable results up to $\Delta t = 2.0\,s$. We ran the same test for both the level set version and the front tracking version of our method. The results demonstrate good stability up to $\Delta t = 3.0\,s$ for both the level set version and the front tracking version. We note that the stability we obtained here is not just limited by our surface tension model but also by the CFL condition imposed by advection. To demonstrate that, we ran another set of tests using the semi-Lagrangian advection, and we obtain stable results up to $\Delta t = 7.0\,s$ for both the level set version and the front tracking version.

6.4.5 Rising Bubble–Convergence

This example tests the convergence behavior under grid refinement concerning a rising bubble. We use the parameters of [86]. A $[-.01\,m, .01\,m] \times [-.02\,m, .01\,m]$ domain is initially filled with water except for a circular air bubble of radius $\frac{1}{300}\,m$ centered at the origin. The fluid parameters are: $g = -9.8\,m\,s^{-2}$, $\sigma = .0728\,kg\,s^{-2}$, $\rho_{water} = 1000\,kg\,m^{-2}$, and $\rho_{air} = 1.226\,kg\,m^{-2}$. We set the viscosity of both water and air to be $\mu = 1.78 \times 10^{-5}\,kg\,s^{-1}$, since the method presented in this paper cannot handle variable viscosity or viscosity jumps in an implicit manner, and the viscosity plays only a a minor role in the dynamics of this example. The computations were carried out on meshes of size 40×60, 80×120, 160×240 and 320×480. Figure 6.4 and Figure 6.5 show snapshot results at different times for the level set version and the front tracking version respectively. We tested the convergence of the interface location

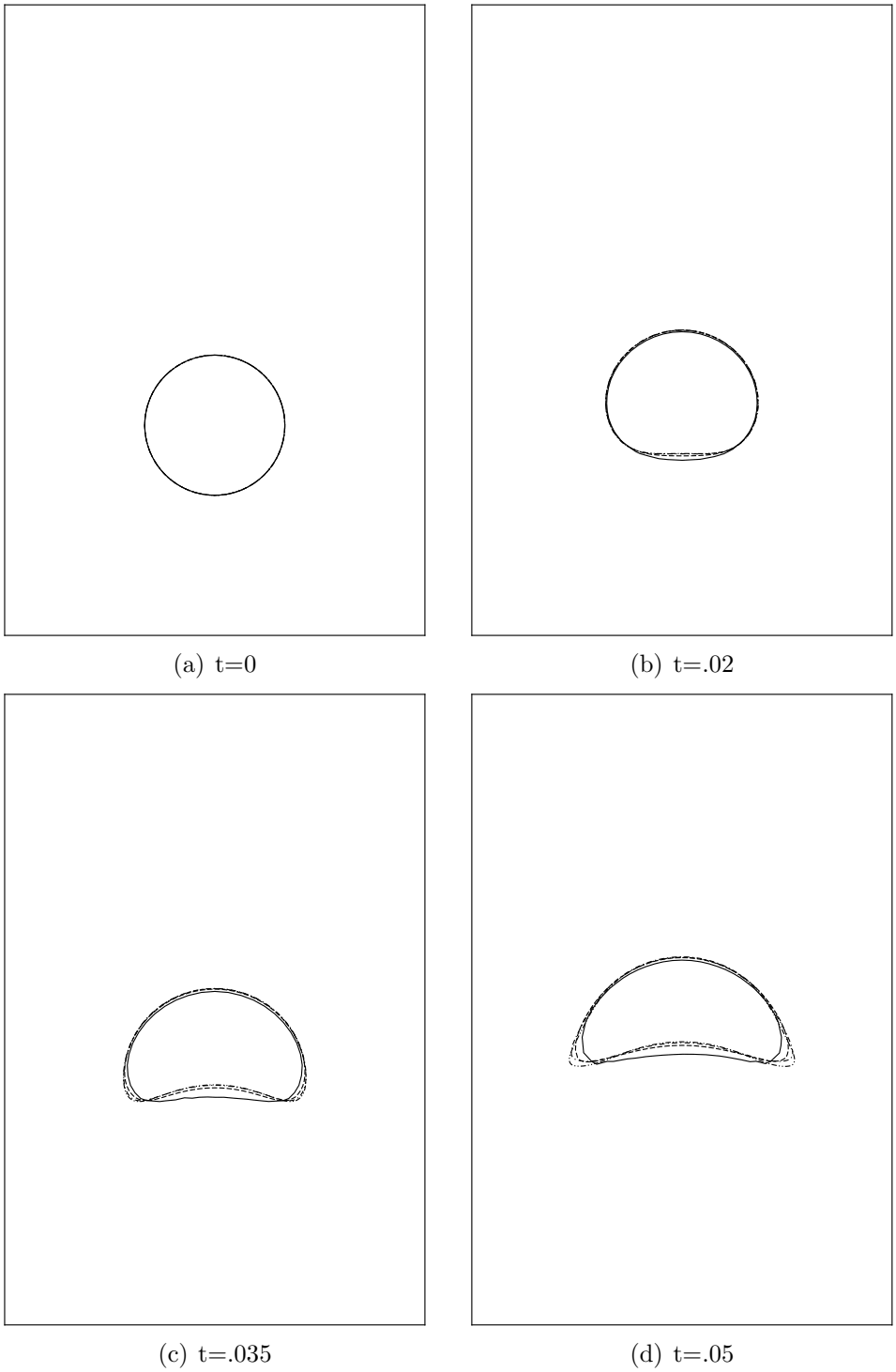


(a) t=0 (b) t=.02



(c) t=.035 (d) t=.05

Figure 6.4: Rising bubble tests for convergence under grid refinement using the level set method. The grid resolutions are: solid line, 40×60; dash line, 80×120; dash-dot line, 160×240; and dash-dot-dot line, 320×480.



(a) t=0 (b) t=.02

(c) t=.035 (d) t=.05

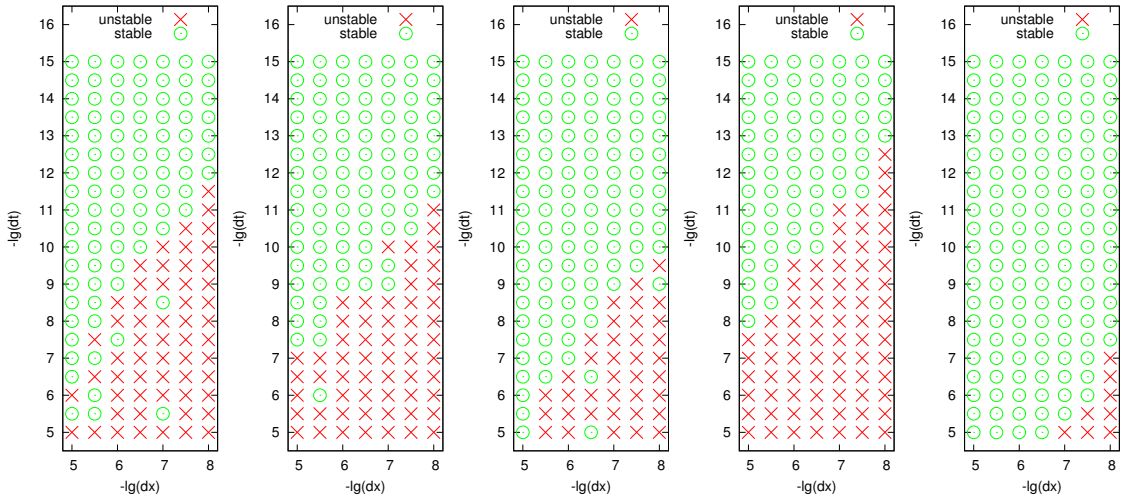
Figure 6.5: Rising bubble tests for convergence under grid refinement using the front tracking method. Grid resolutions: solid line, 40×60; dash line, 80×120; dash-dot line, 160×240; and dash-dot-dot line, 320×480.

at the top middle and bottom middle and in each case observed convergence consistent with first order.

6.4.6 Stability

In this example, we probe the stability characteristics of different integration schemes to determine the effectiveness of our formulation for implicit surface tension at improving stability. In each case, a $[0\,m, 0.04\,m] \times [0\,m, 0.04\,m]$ domain was used. The fluid density is $1000\,kg\,m^{-2}$, and the surface tension coefficient is $\sigma = .0728\,kg\,ms^{-2}$. Since the purpose of this test is to stress the stability of each scheme, we do not use viscosity. The fluid starts in a circle of radius $0.01\,m$ centered within the domain. We test five schemes. The first scheme is a pressure jump scheme based on [86] which we use for comparison purposes. The other four are variants on the schemes proposed here. Two have a front-tracked interface, and the other two have a level set interface. Each interface type is run explicitly (without force derivatives) and implicitly as discussed in Section 6.3.2. Each scheme is run with a wide range of Δx and Δt choices, and in each case it was noted whether the resulting simulation was stable or unstable when run for 500 time steps. The results are shown in Figure 6.6. The pressure jump scheme and the explicit versions of the proposed scheme have similar stability characteristics. The implicit level set version is noticeably more stable than the explicit schemes allowing a Δt approximately 2-3 times larger. The implicit front-tracked version is far more stable, allowing a Δt approximately an order of magnitude larger. Unlike the implicit level set scheme, the position update used on the front-tracked interface corresponds to a backward Euler update, whereas the level set version has a level set update that differs significantly from the update suggested by the force derivatives. Thus, it is not surprising to see better stability from the front-tracked scheme. This opens up an avenue of research to produce an implicit level set scheme, which may be used to improve the stability of the proposed method when the interface is tracked with a level set.

We repeated this on a dynamic configuration. This setup is identical to the first,



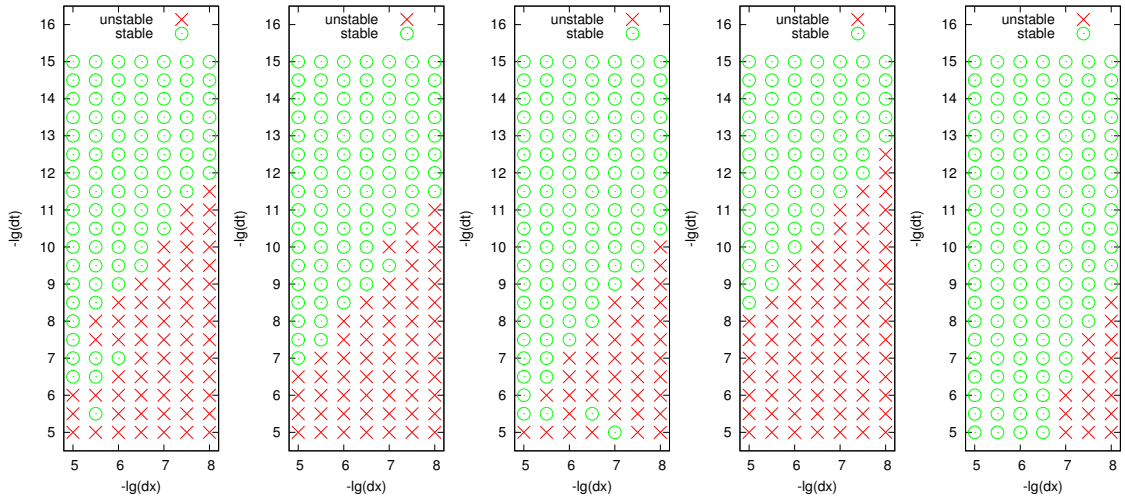
(a) Pressure jump (b) Level set, ex (c) Level set, im (d) Front track, ex (e) Front track, im

Figure 6.6: Stability tests for five schemes on stationary circle. The x axis indicates the Δx used, with more refined simulations on the right. The y axis indicates the Δt used, with smaller time steps at the top. A circle indicates a stable simulation, and an X indicates an unstable simulation. The "ex" and "im" indicate whether the scheme was explicit or implicit.

except that the initial fluid configuration is an ellipse with major axis $0.011\,m$ and minor axis $0.009\,m$. The results are shown in Figure 6.7. The results are quite similar to the stationary circle case, though the dynamic nature of the simulation and the special circle configuration do appear to make a noticeable difference for the implicit front tracking scheme. Since this example has a nonzero velocity as part of its correct behavior, there will be a CFL restriction for advection, but all of the methods became unstable before this CFL restriction was reached.

6.5 Conclusions and Future Work

We have presented a method for applying implicit forces on a Lagrangian mesh to an Eulerian discretization of the Navier Stokes equations. The method leverages the SPD FSI framework of [124] to produce a sparse symmetric positive definite system. The



(a) Pressure jump (b) Level set, ex (c) Level set, im (d) Front track, ex (e) Front track, im

Figure 6.7: Stability tests for five schemes on an ellipse. The x axis indicates the Δx used, with more refined simulations on the right. The y axis indicates the Δt used, with smaller time steps at the top. A circle indicates a stable simulation, and an X indicates an unstable simulation. The "ex" and "im" indicate whether the scheme was explicit or implicit.

resulting method has implicit and fully coupled viscosity, pressure, and Lagrangian forces.

We apply our new framework for forces on a Lagrangian mesh to the case of a surface tension force. We describe two approaches for constructing and maintaining a suitable Lagrangian surface mesh with sufficient accuracy for accurate second derivatives at the surface. We construct a discretization that is able to reduce the magnitude of surface tension parasitic currents down to levels comparable to other state-of-the-art schemes. Finally, we demonstrate the accuracy and enhanced stability of our new method.

Our surface tension discretization is only first order. Nevertheless, many parts are already at second order, and other parts share many attributes with second order methods, including the use of subcell information. We leave it for future work to construct a fully second order accurate discretization.

The stability results in Section 6.4.6 demonstrate drastic improvement in stability for the front tracking interface but only modest improvement for the level set based interface. This is likely because we update the front-tracked interface in a way that is implicit and similar to backward Euler, whereas the level set version is updated instead using an explicit particle level set evolution. This suggests that significant stability improvements could be made by making the level set update implicit in a way compatible with the Lagrangian surface velocities obtained from the coupled system, and we leave this for future work.

There are many interesting avenues for extending the proposed method, which we leave for future work. One may extend the method to treat surfactants [148, 93, 82] and foam [90]. Another potential extension is to problems where a jump in pressure is caused by phenomena other than surface tension, such as an electric potential [160]. Another potential application is to solving partial differential equations on a surfaces as in [18].

Appendix A

Higher Order Orientation Evolution

Second order Runge Kutta has two stages,

1. $\mathbf{R}^{n+\frac{1}{2}} = e^{\Delta t(\boldsymbol{\omega}^n)^*}\mathbf{R}^n$

2. $\mathbf{R}^{n+1} = e^{\Delta t(\boldsymbol{\omega}^{n+\frac{1}{2}})^*}\mathbf{R}^n$

We want to approximate compute $\mathbf{u} = \boldsymbol{\omega}^{n+\frac{1}{2}}$. Note that $\mathbf{R}^{n+\frac{1}{2}} = e^{\frac{\Delta t}{2}\boldsymbol{\omega}^*}\mathbf{R} \approx \mathbf{R} + \frac{1}{2}\Delta t\boldsymbol{\omega}^*\mathbf{R}$. Then, ignoring terms higher than first order,

$$
\begin{aligned}
\mathbf{u} &= (\mathbf{I}^{-1})^{n+\frac{1}{2}}\mathbf{L} \\
\mathbf{u} &= (\mathbf{R}^{n+\frac{1}{2}})\mathbf{I}_0^{-1}(\mathbf{R}^{n+\frac{1}{2}})^T\mathbf{L} \\
\mathbf{u} &= (\mathbf{R} + \frac{1}{2}\Delta t\boldsymbol{\omega}^*\mathbf{R})\mathbf{I}_0^{-1}(\mathbf{R} + \frac{1}{2}\Delta t\boldsymbol{\omega}^*\mathbf{R})^T\mathbf{L} \\
\mathbf{u} &= \mathbf{R}\mathbf{I}_0^{-1}\mathbf{R}^T\mathbf{L} + \frac{1}{2}\Delta t(\boldsymbol{\omega}^*\mathbf{R})\mathbf{I}_0^{-1}\mathbf{R}^T\mathbf{L} + \frac{1}{2}\Delta t\mathbf{R}\mathbf{I}_0^{-1}(\boldsymbol{\omega}^*\mathbf{R})^T\mathbf{L} \\
\mathbf{u} &= \boldsymbol{\omega} + \frac{1}{2}\Delta t\boldsymbol{\omega}^*\boldsymbol{\omega} + \frac{1}{2}\Delta t\mathbf{I}^{-1}(\boldsymbol{\omega}^*)^T\mathbf{L} \\
\mathbf{u} &= \boldsymbol{\omega} + \frac{1}{2}\Delta t\mathbf{I}^{-1}\mathbf{L}^*\boldsymbol{\omega}
\end{aligned}
$$

This is the second order update rule. Note that the exponential is used on the final update to guarantee that a rotation matrix is produced. This restriction is effectively relaxed for the first stage, since the final rotation will be a rotation even if the first one is not.

Deriving the third order method is not as easy, since the third order Runge Kutta schemes end with an averaging step, which is problematic with rotations. Instead, it can be derived directly from a Taylor series expansion. First it is necessary to compute some derivatives. Note that in the absence of forces, $\dot{\mathbf{L}} = \mathbf{0}$. The inertia tensor in the body's reference configuration is also fixed, so that $(\mathbf{I}_0^{-1})' = \mathbf{0}$. The identity matrix is denoted as $\boldsymbol{\delta}$ here to distinguish it from the inertia tensor $\mathbf{I}$.

$$
\begin{aligned}
\mathbf{R}\mathbf{R}^T &= \boldsymbol{\delta} \\
\dot{\mathbf{R}} &= \boldsymbol{\omega}^*\mathbf{R} \\
(\mathbf{I}^{-1})' &= (\mathbf{R}\mathbf{I}_0^{-1}\mathbf{R}^T)' = \dot{\mathbf{R}}\mathbf{I}_0^{-1}\mathbf{R}^T + \mathbf{R}\mathbf{I}_0^{-1}\dot{\mathbf{R}}^T = \boldsymbol{\omega}^*\mathbf{R}\mathbf{I}_0^{-1}\mathbf{R}^T + \mathbf{R}\mathbf{I}_0^{-1}\mathbf{R}^T(\boldsymbol{\omega}^*)^T \\
&= \boldsymbol{\omega}^*\mathbf{I}^{-1} - \mathbf{I}^{-1}\boldsymbol{\omega}^* \\
\dot{\boldsymbol{\omega}} &= (\mathbf{I}^{-1}\mathbf{L})' = (\boldsymbol{\omega}^*\mathbf{I}^{-1} - \mathbf{I}^{-1}\boldsymbol{\omega}^*)\mathbf{L} = \mathbf{I}^{-1}\mathbf{L}^*\boldsymbol{\omega} \\
\ddot{\mathbf{R}} &= (\boldsymbol{\omega}^*\mathbf{R})' = \dot{\boldsymbol{\omega}}^*\mathbf{R} + \boldsymbol{\omega}^*\dot{\mathbf{R}} = \dot{\boldsymbol{\omega}}^*\mathbf{R} + \boldsymbol{\omega}^*\boldsymbol{\omega}^*\mathbf{R} \\
\dddot{\mathbf{R}} &= (\dot{\boldsymbol{\omega}}^*\mathbf{R} + \boldsymbol{\omega}^*\boldsymbol{\omega}^*\mathbf{R})' = \ddot{\boldsymbol{\omega}}^*\mathbf{R} + \dot{\boldsymbol{\omega}}^*\dot{\mathbf{R}} + \dot{\boldsymbol{\omega}}^*\boldsymbol{\omega}^*\mathbf{R} + \boldsymbol{\omega}^*\dot{\boldsymbol{\omega}}^*\mathbf{R} + \boldsymbol{\omega}^*\boldsymbol{\omega}^*\dot{\mathbf{R}} \\
&= \ddot{\boldsymbol{\omega}}^*\mathbf{R} + 2\dot{\boldsymbol{\omega}}^*\boldsymbol{\omega}^*\mathbf{R} + \boldsymbol{\omega}^*\dot{\boldsymbol{\omega}}^*\mathbf{R} + \boldsymbol{\omega}^*\boldsymbol{\omega}^*\boldsymbol{\omega}^*\mathbf{R}
\end{aligned}
$$

Next, consider the update rule, ignoring fourth order terms

$$
\begin{aligned}
\mathbf{R}^{n+1}\mathbf{R}^T &= e^{\Delta t\mathbf{u}_1^* + \Delta t^2\mathbf{u}_2^* + \Delta t^3\mathbf{u}_3^*} \\
&= \boldsymbol{\delta} + \Delta t(\mathbf{u}_1^* + \Delta t\mathbf{u}_2^* + \Delta t^2\mathbf{u}_3^*) + \frac{1}{2}\Delta t^2(\mathbf{u}_1^*\mathbf{u}_1^* + \Delta t\mathbf{u}_2^*\mathbf{u}_1^* + \Delta t\mathbf{u}_1^*\mathbf{u}_2^*) \\
&\quad + \frac{1}{6}\Delta t^3\mathbf{u}_1^*\mathbf{u}_1^*\mathbf{u}_1^* \\
&= \boldsymbol{\delta} + \Delta t\mathbf{u}_1^* + \frac{1}{2}\Delta t^2(\mathbf{u}_1^*\mathbf{u}_1^* + 2\mathbf{u}_2^*) + \frac{1}{6}\Delta t^3(\mathbf{u}_1^*\mathbf{u}_1^*\mathbf{u}_1^* + 3\mathbf{u}_2^*\mathbf{u}_1^* + 3\mathbf{u}_1^*\mathbf{u}_2^* + 6\mathbf{u}_3^*)
\end{aligned}
$$

Next, consider the Taylor series expansion of $\mathbf{R}^{n+1}\mathbf{R}^T$

$$\begin{aligned}\mathbf{R}^{n+1}\mathbf{R}^T &= (\mathbf{R} + \Delta t\dot{\mathbf{R}} + \frac{1}{2}\Delta t^2\ddot{\mathbf{R}} + \frac{1}{6}\Delta t^3\dddot{\mathbf{R}})\mathbf{R}^T \\ &= \boldsymbol{\delta} + \Delta t\boldsymbol{\omega}^* + \frac{1}{2}\Delta t^2(\dot{\boldsymbol{\omega}}^* + \boldsymbol{\omega}^*\boldsymbol{\omega}^*) + \frac{1}{6}\Delta t^3(\ddot{\boldsymbol{\omega}}^* + 2\dot{\boldsymbol{\omega}}^*\boldsymbol{\omega}^* + \boldsymbol{\omega}^*\dot{\boldsymbol{\omega}}^* + \boldsymbol{\omega}^*\boldsymbol{\omega}^*\boldsymbol{\omega}^*)\end{aligned}$$

The identity terms match, and the first order terms imply $\mathbf{u}_1 = \boldsymbol{\omega}$. Next, consider the second order terms

$$\begin{aligned}\mathbf{u}_1^*\mathbf{u}_1^* + 2\mathbf{u}_2^* &= \dot{\boldsymbol{\omega}}^* + \boldsymbol{\omega}^*\boldsymbol{\omega}^* \\ \mathbf{u}_2 &= \frac{1}{2}\dot{\boldsymbol{\omega}}\end{aligned}$$

These together with $\dot{\boldsymbol{\omega}} = \mathbf{I}^{-1}\mathbf{L}^*\boldsymbol{\omega}$ reproduce the second order evolution result. Finally, match up the third order terms.

$$\begin{aligned}\mathbf{u}_1^*\mathbf{u}_1^*\mathbf{u}_1^* + 3\mathbf{u}_2^*\mathbf{u}_1^* + 3\mathbf{u}_1^*\mathbf{u}_2^* + 6\mathbf{u}_3^* &= \ddot{\boldsymbol{\omega}}^* + 2\dot{\boldsymbol{\omega}}^*\boldsymbol{\omega}^* + \boldsymbol{\omega}^*\dot{\boldsymbol{\omega}}^* + \boldsymbol{\omega}^*\boldsymbol{\omega}^*\boldsymbol{\omega}^* \\ \mathbf{u}_3^* &= \frac{1}{6}\ddot{\boldsymbol{\omega}}^* + \frac{1}{12}(\dot{\boldsymbol{\omega}}^*\boldsymbol{\omega}^* - \boldsymbol{\omega}^*\dot{\boldsymbol{\omega}}^*)\end{aligned}$$

The last term is not in a suitable form, but this can be fixed with Jacobi's identity.

$$\begin{aligned}\mathbf{x}\times(\mathbf{y}\times\mathbf{z}) + \mathbf{z}\times(\mathbf{x}\times\mathbf{y}) + \mathbf{y}\times(\mathbf{z}\times\mathbf{x}) &= \mathbf{0} \\ \mathbf{x}\times(\mathbf{y}\times\mathbf{z}) + \mathbf{z}\times(\mathbf{x}\times\mathbf{y}) - \mathbf{y}\times(\mathbf{x}\times\mathbf{z}) &= \mathbf{0} \\ \mathbf{x}\times(\mathbf{y}\times\mathbf{z}) - \mathbf{y}\times(\mathbf{x}\times\mathbf{z}) &= (\mathbf{x}\times\mathbf{y})\times\mathbf{z} \\ \mathbf{x}^*\mathbf{y}^*\mathbf{z} - \mathbf{y}^*\mathbf{x}^*\mathbf{z} &= (\mathbf{x}^*\mathbf{y})^*\mathbf{z} \\ \mathbf{x}^*\mathbf{y}^* - \mathbf{y}^*\mathbf{x}^* &= (\mathbf{x}^*\mathbf{y})^*\end{aligned}$$

Now, the third order terms become

$$\begin{aligned}
\mathbf{u}_3^* &= \frac{1}{6}\ddot{\boldsymbol{\omega}}^* + \frac{1}{12}(\dot{\boldsymbol{\omega}}^*\boldsymbol{\omega}^* - \boldsymbol{\omega}^*\dot{\boldsymbol{\omega}}^*) \\
\mathbf{u}_3^* &= \frac{1}{6}\ddot{\boldsymbol{\omega}}^* + \frac{1}{12}(\dot{\boldsymbol{\omega}}^*\boldsymbol{\omega})^* \\
\mathbf{u}_3 &= \frac{1}{6}\ddot{\boldsymbol{\omega}} + \frac{1}{12}\dot{\boldsymbol{\omega}}^*\boldsymbol{\omega} \\
\mathbf{u}_3 &= \frac{1}{6}\ddot{\boldsymbol{\omega}} - \frac{1}{12}\boldsymbol{\omega}^*\dot{\boldsymbol{\omega}} \\
\mathbf{u}_3 &= \frac{1}{6}(\mathbf{I}^{-1}\mathbf{L}^*\boldsymbol{\omega})' - \frac{1}{12}\boldsymbol{\omega}^*\mathbf{I}^{-1}\mathbf{L}^*\boldsymbol{\omega} \\
\mathbf{u}_3 &= \frac{1}{6}(\mathbf{I}^{-1})'\mathbf{L}^*\boldsymbol{\omega} + \frac{1}{6}\mathbf{I}^{-1}\mathbf{L}^*\dot{\boldsymbol{\omega}} - \frac{1}{12}\boldsymbol{\omega}^*\mathbf{I}^{-1}\mathbf{L}^*\boldsymbol{\omega} \\
\mathbf{u}_3 &= \frac{1}{6}(\boldsymbol{\omega}^*\mathbf{I}^{-1} - \mathbf{I}^{-1}\boldsymbol{\omega}^*)\mathbf{L}^*\boldsymbol{\omega} + \frac{1}{6}\mathbf{I}^{-1}\mathbf{L}^*\mathbf{I}^{-1}\mathbf{L}^*\boldsymbol{\omega} - \frac{1}{12}\boldsymbol{\omega}^*\mathbf{I}^{-1}\mathbf{L}^*\boldsymbol{\omega} \\
\mathbf{u}_3 &= \frac{1}{12}\boldsymbol{\omega}^*\mathbf{I}^{-1}\mathbf{L}^*\boldsymbol{\omega} - \frac{1}{6}\mathbf{I}^{-1}\boldsymbol{\omega}^*\mathbf{L}^*\boldsymbol{\omega} + \frac{1}{6}\mathbf{I}^{-1}\mathbf{L}^*\mathbf{I}^{-1}\mathbf{L}^*\boldsymbol{\omega}
\end{aligned}$$

This completes the third order correction.

Appendix B

Interpenetration Resolution

B.1 Introduction

We process interpenetration resolution on a *central* body, whose quantities are denoted with subscript c, and the *outer* bodies intersecting it, denoted with subscript d for a dynamic body, s for a static body, or o for either type. Let D denote the set of dynamic outer bodies, S denote the set of static outer bodies, and $O = D \cup S$. Let $\mathbf{p}_o$ be the deepest intersection point between body c and body o. Let $\mathbf{d}_o$ be the vector from the point on the surface of body c to the corresponding point on the surface of body o. Interpenetration resolution tries to make these two points coincident by creating a relative velocity $\mathbf{v}_r$ that will be evolved for an arbitrary pseudo-time $\Delta\tau$. The desired relative velocity between bodies c and o at the point $\mathbf{p}_o$ is

$$\mathbf{v}_r = \mathbf{d}_o \Delta\tau^{-1}.$$

Note that this velocity is only used to determine how far the bodies should be displaced. The momenta of the bodies are unchanged. Note that since the orientation of bodies may be changed, leaving the angular momentum unchanged is different

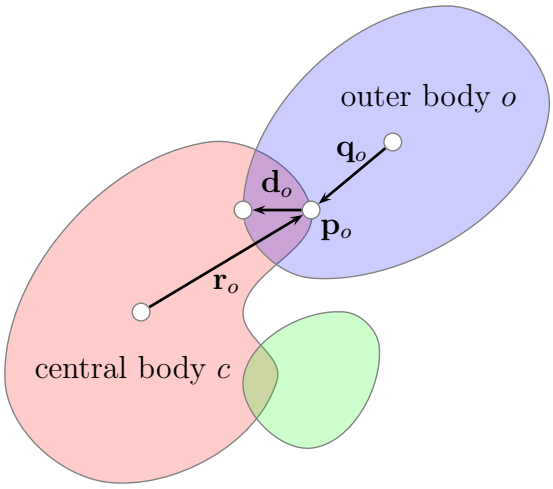


Figure B.1: Central body intersecting two outer bodies.

from leaving the angular velocity unchanged. We choose to leave angular momentum unchanged since it is the conserved quantity and recompute the angular velocity afterwards.

Let $\mathbf{j}_o$ be the impulse applied to outer body o at the intersection point $\mathbf{p}_o$. Then the total linear impulse applied to the central body is

$$\mathbf{j} = -\sum_{o \in O} \mathbf{j}_o,$$

and the total angular impulse applied to the central body is

$$\mathbf{j}_\tau = -\sum_{o \in O} \mathbf{r}_o^* \mathbf{j}_o,$$

where $\mathbf{r}_o$ is the vector from the center of mass of body c to $\mathbf{p}_o$. The impulse factor for a dynamic body d measured at location $\mathbf{p}_d$ is given by

$$\mathbf{K}_d = m_d^{-1}\boldsymbol{\delta} + \mathbf{q}_d^T \mathbf{I}_d^{-1} \mathbf{q}_d,$$

where $\mathbf{q}_d$ is the vector from the center of mass of body d to $\mathbf{p}_d$. The impulse factor is the matrix such that for a chosen displacement $\mathbf{r}_d$ from the center of mass of the

rigid body, an impulse $\mathbf{j}_d$ results in a velocity change of $\Delta\mathbf{v}_d = \mathbf{K}_d\mathbf{j}_d$. That is, $\mathbf{K}_d^{-1}$ is a sort of effective mass. This 3×3 matrix is sufficient, and the full 6×6 version derived in Appendix C is not required. This is because impulses are applied but not torques, and the relative velocity at point of torque is required but not the angular velocity.

Static bodies are the limit of infinite mass, and the impulse factor is zero in this limit. If there is more than one static body, it is not sufficient to consider static bodies in this way. If more than one surrounding body has infinite mass, the central body may be stuck between two such bodies. In this case, it may be that no suitable correction exists, and instead one effectively considers a slightly relaxed problem. To do this, it is useful to consider a notion of how the bodies go to infinity relative to each other. We take the mass and inertia tensor of body s to be $\epsilon^{-1}m_s$ and $\epsilon^{-1}\mathbf{I}_s$, where $\epsilon \to 0$ is the limit of the body to being static. In particular, all static bodies are taken to infinity together. The static body s has the impulse factor

$$\mathbf{K}_s = \epsilon\hat{\mathbf{K}}_s = \epsilon(m_s^{-1}\boldsymbol{\delta} + \mathbf{q}_s^{*T}\mathbf{I}_s^{-1}\mathbf{q}_s^*).$$

Note that even though a nonzero impulse factor is being computed for the static body, it is still regarded as static and is never moved.

B.2 Solving for the Central Body

We wish to apply an impulses $\mathbf{j}_o$ at each of the penetration points $\mathbf{p}_o$ such that the central body and each outer body satisfy

$$m_c^{-1}\mathbf{j} + \mathbf{r}_o^{*T}\mathbf{I}_c^{-1}\mathbf{j}_\tau - \mathbf{K}_o\mathbf{j}_o = \mathbf{v}_r. \tag{B.1}$$

That is, the impulse should cause a relative velocity sufficient to resolve the interpenetration. The interesting thing about this equation is that once the impulse $\mathbf{j}$ and torque $\mathbf{j}_\tau$ on the central body have been computed, the individual impulses $\mathbf{j}_o$ for

the outer bodies can be computed directly. This allows us to eliminate all of these impulses $\mathbf{j}_o$ from the system and solve for $\mathbf{j}$ and $\mathbf{j}_\tau$ efficiently as a 6×6 system. Solving for $\mathbf{j}_o$, we have

$$\mathbf{j}_o = \mathbf{K}_o^{-1}\big(-\mathbf{v}_r + m_c^{-1}\mathbf{j} + \mathbf{r}_o^{*T}\mathbf{I}_c^{-1}\mathbf{j}_\tau\big)$$

Summing over all the bodies in O, we have

$$\mathbf{j} = -\sum_o \mathbf{j}_o = \left(\sum_o \mathbf{K}_o^{-1}\mathbf{v}_r\right) - \left(\sum_o \mathbf{K}_o^{-1}\right) m_c^{-1}\mathbf{j} - \left(\sum_o \mathbf{K}_o^{-1}\mathbf{r}_o^{*T}\right)\mathbf{I}_c^{-1}\mathbf{j}_\tau$$
$$\mathbf{j}_\tau = -\sum_o \mathbf{r}_o^{*}\mathbf{j}_o = \left(\sum_o \mathbf{r}_o^{*}\mathbf{K}_o^{-1}\mathbf{v}_r\right) - \left(\sum_o \mathbf{r}_o^{*}\mathbf{K}_o^{-1}\right) m_c^{-1}\mathbf{j} - \left(\sum_o \mathbf{r}_o^{*}\mathbf{K}_o^{-1}\mathbf{r}_o^{*T}\right)\mathbf{I}_c^{-1}\mathbf{j}_\tau.$$

Then, we can write the system

$$\begin{pmatrix} m_c\boldsymbol{\delta} + \sum_o \mathbf{K}_o^{-1} & \sum_o \mathbf{K}_o^{-1}\mathbf{r}_o^{*T} \\ \sum_o \mathbf{r}_o^{*}\mathbf{K}_o^{-1} & \mathbf{I}_c + \sum_o \mathbf{r}_o^{*}\mathbf{K}_o^{-1}\mathbf{r}_o^{*T} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{v} \\ \Delta\boldsymbol{\omega} \end{pmatrix} = \begin{pmatrix} \sum_o \mathbf{K}_o^{-1}\mathbf{v}_r \\ \sum_o \mathbf{r}_o^{*}\mathbf{K}_o^{-1}\mathbf{v}_r \end{pmatrix}, \tag{B.2}$$

where $\Delta\mathbf{v} = m_c^{-1}\mathbf{j}$ and $\Delta\boldsymbol{\omega} = \mathbf{I}_c^{-1}\mathbf{j}_\tau$. Finally, the impulses of the outer bodies are obtained by

$$\mathbf{j}_o = \mathbf{K}_o^{-1}\big(-\mathbf{v}_r + \Delta\mathbf{v} + \mathbf{r}_o^{*T}\Delta\boldsymbol{\omega}\big). \tag{B.3}$$

B.2.1 Applying the push

For static outer bodies s, no push is applied. For each dynamic outer body d, we apply the push as

$$\mathbf{x}_d^{new} = \mathbf{x}_d + \Delta\tau m_d^{-1}\mathbf{j}_d \qquad \mathbf{R}_d^{new} = e^{\Delta\tau(\mathbf{I}_d^{-1}\mathbf{r}_d^{*}\mathbf{j}_d)^{*}}\mathbf{R}_d.$$

If the central body is dynamic, the push is applied to the central body as

$$\mathbf{x}_c^{new} = \mathbf{x}_c + \Delta\tau\Delta\mathbf{v} \qquad \mathbf{R}_c^{new} = e^{\Delta\tau\Delta\boldsymbol{\omega}^{*}}\mathbf{R}_c.$$

Note that, due to the non-dynamic nature of the interpenetration resolution process, these equations appear a bit peculiar. The actual velocities of the bodies are never used.

B.3 Special Configuration Cases

The way in which the impulses are computed differs depending on how many static bodies are involved. Two cases are straightforward. If the central body and all of the outer bodies are dynamic, then (B.2) is symmetric positive definite and can be uniquely solved for $\mathbf{j}$ and $\mathbf{j}_\tau$ (or alternatively $\Delta\mathbf{v}$ and $\Delta\boldsymbol{\omega}$). Then, $\mathbf{j}_d$ is obtained from (B.3).

A second easy case occurs if the central body is static. That is, $m_c^{-1} = 0$ and $\mathbf{I}_c^{-1} = 0$. Then, (B.3) becomes

$$\mathbf{j}_d = -\mathbf{K}_d^{-1}\mathbf{v}_d.$$

Thus we solve for each $\mathbf{j}_d$ independently and apply it to the body d as in section B.2.1. Note that no push is applied to the central body as it is static, and $\Delta\mathbf{x}_c = \mathbf{0}$, $\Delta\mathbf{R}_c = \mathbf{0}$. Outer static bodies can be ignored in this case, since it makes no sense to exchange an impulse between two bodies that will ignore that impulse.

There remaining cases occur when the central body is dynamic but at least one outer body is static. When this occurs, the constraints involving static bodies are effectively stronger than those with dynamic bodies. This is because the movement of the central body may be mostly determined by what is required to resolve the penetration due to the static body. Constraints with dynamic outer bodies are easier to resolve, since the outer body may move as well. The character of the interaction between the constraints imposed by the static bodies is different when there is just one outer body from when more than one is present.

B.3.1 One Static Outer Body

When only one outer body is static, its mass can be treated as infinite. Three of the six total degrees of freedom for the central body are determined by the impulse required to correct interpenetration with the static outer body. The other three are determined by the dynamic bodies. Even though half of the degrees of freedom are effectively dictated by the static outer body, the contribution of both the static and dynamic bodies to the six degrees of freedom must be distinguished. This makes static outer body cases more difficult than the purely dynamic cases.

The static outer body s has $\mathbf{K}_s = 0$. Then, (B.1) gives

$$\Delta\mathbf{v} + \mathbf{r}_s^{*T}\Delta\boldsymbol{\omega} = \mathbf{v}_s. \tag{B.4}$$

Starting with the system in (B.2), left multiply the first equation by $\mathbf{r}_s^*$ and subtract from the second equation to get

$$\left(\sum_o \mathbf{r}_o^*\mathbf{K}_o^{-1} - m_c\mathbf{r}_s^* - \sum_o \mathbf{r}_s^*\mathbf{K}_o^{-1}\right)\Delta\mathbf{v} + \left(\mathbf{I}_c + \sum_o \mathbf{r}_o^*\mathbf{K}_o^{-1}\mathbf{r}_o^{*T} - \sum_o \mathbf{r}_s^*\mathbf{K}_o^{-1}\mathbf{r}_o^{*T}\right)\Delta\boldsymbol{\omega}$$

$$= \sum_o \mathbf{r}_o^*\mathbf{K}_o^{-1}\mathbf{v}_o - \sum_o \mathbf{r}_o^*\mathbf{K}_o^{-1}\mathbf{v}_o.$$

Note that the terms with body s cancel, and we exclude them explicitly. Letting $\tilde{\mathbf{r}}_d^* = \mathbf{r}_d^* - \mathbf{r}_s^*$ and recalling that d indicates outer bodies that are not static, we have

$$\left(\sum_d \tilde{\mathbf{r}}_d^*\mathbf{K}_d^{-1} - m_c\mathbf{r}_s^*\right)\Delta\mathbf{v} + \left(\mathbf{I}_c + \sum_d \tilde{\mathbf{r}}_d^*\mathbf{K}_d^{-1}\mathbf{r}_d^{*T}\right)\Delta\boldsymbol{\omega} = \sum_d \tilde{\mathbf{r}}_d^*\mathbf{K}_d^{-1}\mathbf{v}_d. \tag{B.5}$$

Subtracting $\sum_d \tilde{\mathbf{r}}_d^*\mathbf{K}_d^{-1}$ times equation (B.4) from equation (B.5) gives the system

$$\begin{pmatrix} \boldsymbol{\delta} & \mathbf{r}_s^{*T} \\ -m_c\mathbf{r}_s^* & \mathbf{I}_c + \sum_d \tilde{\mathbf{r}}_d^*\mathbf{K}_d^{-1}\mathbf{r}_d^{*T} - \sum_d \tilde{\mathbf{r}}_d^*\mathbf{K}_d^{-1}\mathbf{r}_s^{*T} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{v} \\ \Delta\boldsymbol{\omega} \end{pmatrix} = \begin{pmatrix} \mathbf{v}_s \\ \sum_d \tilde{\mathbf{r}}_d^*\mathbf{K}_d^{-1}\mathbf{v}_d - \sum_d \tilde{\mathbf{r}}_d^*\mathbf{K}_d^{-1}\mathbf{v}_s \end{pmatrix}$$

Substituting $\tilde{\mathbf{v}}_d = \mathbf{v}_d - \mathbf{v}_s$, we have

$$\begin{pmatrix} \boldsymbol{\delta} & \mathbf{r}_s^{*T} \\ -m_c\mathbf{r}_s^* & \mathbf{I}_c + \sum_d \tilde{\mathbf{r}}_d^*\mathbf{K}_d^{-1}\tilde{\mathbf{r}}_d^{*T} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{v} \\ \Delta\boldsymbol{\omega} \end{pmatrix} = \begin{pmatrix} \mathbf{v}_s \\ \sum_d \tilde{\mathbf{r}}_d^*\mathbf{K}_d^{-1}\tilde{\mathbf{v}}_d \end{pmatrix}$$

Performing a step of Gaussian elimination and negating the second equation yields

$$\begin{pmatrix} \boldsymbol{\delta} & \mathbf{r}_s^{*T} \\ \mathbf{0} & \mathbf{I}_c + \sum_d \tilde{\mathbf{r}}_d^*\mathbf{K}_d^{-1}\tilde{\mathbf{r}}_d^{*T} + m_c\mathbf{r}_s^*\mathbf{r}_s^{*T} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{v} \\ \Delta\boldsymbol{\omega} \end{pmatrix} = \begin{pmatrix} \mathbf{v}_s \\ \sum_d \tilde{\mathbf{r}}_d^*\mathbf{K}_d^{-1}\tilde{\mathbf{v}}_d + m_c\mathbf{r}_s^*\mathbf{v}_s \end{pmatrix}$$

Observe that both block diagonal terms are strictly positive definite, so that the matrix equation is nonsingular and possesses a unique solution. In this case, only a 3×3 system must be solved.

B.3.2 Any Number of Static Bodies

The final case is a degenerate case. When more than one outer body is infinitely massive, the constraints will compete. We deal with this by considering the bodies to be finitely massive and consider the limit as their masses increase to infinity together. As before, the degrees of freedom are partially determined by the static outer bodies, and the number of degrees of freedom chosen by them depends on how many there are and what their spatial configuration is.

As before $\mathbf{K}_s = \epsilon\hat{\mathbf{K}}_s$ where $\epsilon \to 0$. Introducing the definitions

$$\mathbf{A}_d = \begin{pmatrix} m_c\boldsymbol{\delta} + \sum_d \mathbf{K}_d^{-1} & \sum_d \mathbf{K}_d^{-1}\mathbf{r}_d^{*T} \\ \sum_d \mathbf{r}_d^*\mathbf{K}_d^{-1} & \mathbf{I}_c + \sum_d \mathbf{r}_d^*\mathbf{K}_d^{-1}\mathbf{r}_d^{*T} \end{pmatrix} \qquad \mathbf{b}_d = \begin{pmatrix} \sum_d \mathbf{K}_d^{-1}\mathbf{v}_d \\ \sum_d \mathbf{r}_d^*\mathbf{K}_d^{-1}\mathbf{v}_d \end{pmatrix}$$

$$\mathbf{A}_s = \begin{pmatrix} \sum_s \hat{\mathbf{K}}_s^{-1} & \sum_s \hat{\mathbf{K}}_s^{-1}\mathbf{r}_s^{*T} \\ \sum_s \mathbf{r}_s^*\hat{\mathbf{K}}_s^{-1} & \sum_s \mathbf{r}_s^*\hat{\mathbf{K}}_s^{-1}\mathbf{r}_s^{*T} \end{pmatrix} \qquad \mathbf{b}_s = \begin{pmatrix} \sum_s \hat{\mathbf{K}}_s^{-1}\mathbf{v}_s \\ \sum_s \mathbf{r}_s^*\hat{\mathbf{K}}_s^{-1}\mathbf{v}_s \end{pmatrix}$$

(B.2) can be rewritten as

$$\left(\mathbf{A}_d + \epsilon^{-1}\mathbf{A}_s\right)\begin{pmatrix}\Delta\mathbf{v}\\ \Delta\boldsymbol{\omega}\end{pmatrix} = \mathbf{b}_d + \epsilon^{-1}\mathbf{b}_s$$

Multiplying through by ϵ and taking $\epsilon \to 0$, this system becomes

$$\mathbf{A}_s\begin{pmatrix}\Delta\mathbf{v}\\ \Delta\boldsymbol{\omega}\end{pmatrix} = \mathbf{b}_s$$

If $\mathbf{A}_s$ is nonsingular, this system can be solved for the motion of the central body, which can be substituted into (B.3) to yield the full set of impulses. This corresponds to the case when there are enough static outer bodies to entirely determine how the central body must be moved to resolve the interpenetrations. The dynamic outer bodies do not affect the correction applied to the central body.

Otherwise, we must augment this system to get a full system. Consider that $\mathbf{Z}$ is a matrix whose columns form a basis for the nullspace of the symmetric $\mathbf{A}_s$. Let $\mathbf{Y}$ be the matrix basic for the column space of $\mathbf{A}_s$. Consider the products of the full system by these two matrices. Multiplying first by $\mathbf{Z}^T$ and using $\mathbf{Z}^T\mathbf{A}_s = 0$ yields

$$\mathbf{Z}^T\mathbf{A}_d\begin{pmatrix}\Delta\mathbf{v}\\ \Delta\boldsymbol{\omega}\end{pmatrix} = \mathbf{Z}^T(\mathbf{b}_d + \epsilon^{-1}\mathbf{b}_s)$$

The rest comes from the static bodies by instead multiplying by $\epsilon\mathbf{Y}^T$ and taking the limit $\epsilon \to 0$ and is

$$\mathbf{Y}^T\mathbf{A}_s\begin{pmatrix}\Delta\mathbf{v}\\ \Delta\boldsymbol{\omega}\end{pmatrix} = \mathbf{Y}^T\mathbf{b}_s$$

This results in the full-rank 6×6 (3×3 in 2D) system

$$\begin{pmatrix}\mathbf{Y}^T\mathbf{A}_s\\ \mathbf{Z}^T\mathbf{A}_d\end{pmatrix}\begin{pmatrix}\Delta\mathbf{v}\\ \Delta\boldsymbol{\omega}\end{pmatrix} = \begin{pmatrix}\mathbf{Y}^T\mathbf{b}_s\\ \mathbf{Z}^T(\mathbf{b}_d + \epsilon^{-1}\mathbf{b}_s)\end{pmatrix}$$

This right hand side is somewhat problematic, since it contains ϵ^{-1}. In the next section, we will show that $\mathbf{Z}^T\mathbf{b}_s = 0$, so that our system reduces to

$$\begin{pmatrix}\mathbf{Y}^T\mathbf{A}_s \\ \mathbf{Z}^T\mathbf{A}_d\end{pmatrix}\begin{pmatrix}\Delta\mathbf{v} \\ \Delta\boldsymbol{\omega}\end{pmatrix} = \begin{pmatrix}\mathbf{Y}^T\mathbf{b}_s \\ \mathbf{Z}^T\mathbf{b}_d\end{pmatrix}$$

B.3.3 Nullspace Computation

What remains is to find $\mathbf{Z}$ (from which $\mathbf{Y}$ is the complement). This can be computed directly, since the matrix is only 6×6. Being required only in the degenerate case of two or more outer bodies, this may be acceptable. However, it is also possible to deduce the nullspace directly at the cost of a bit more logic. To do this, it is helpful to consider more carefully the physical layout of the static bodies. The cases that are important are if all static bodies intersect the central body at the same point, all intersection points with static bodies are in a straight line, and the general case where they are not collinear. We begin with the collinear case.

Recall that $\mathbf{r}_s$ are the locations of the collision points with the s static bodies. Let $\mathbf{c}$ be the centroid and $\mathbf{u}$ a unit vector parallel to the line of the collision points so that

$$\mathbf{c} = \frac{1}{s}\sum_s \mathbf{r}_s, \qquad \mathbf{r}_s = \mathbf{c} + \alpha_s\mathbf{u}$$

If $\mathbf{C}$ is the matrix whose columns are the vectors $\mathbf{r}_s - c$, then the value u can be determined as the eigenvector corresponding to the only nonzero (to within some tolerance) eigenvalue of the 3×3 matrix $\mathbf{C}\mathbf{C}^T$. (In the case of all eigenvalues roughly zero, all of the collision points coincide. In the case of two or more nonzero eigenvalues, we are in the general case.) If we assume collinearity, we can express

$$\mathbf{C}\mathbf{C}^T = \sum_s (\mathbf{r}_s - c)(\mathbf{r}_s - c)^T = \sum_s \alpha_s^2\mathbf{u}\mathbf{u}^T$$

It is clear that this has rank one. Furthermore,

$$\mathbf{C}\mathbf{C}^T\mathbf{u} = \sum_s \alpha_s^2 \mathbf{u}\mathbf{u}^T\mathbf{u} = \sum_s \alpha_s^2 \mathbf{u}, \qquad \lambda = \sum_s \alpha_s^2$$

Then It follows that the coincident case corresponds to the third eigenvalue being zero as well.

Next, we compute the nullspace $\mathbf{Z}$ of $\mathbf{A}_s$ for the collinear case. This case is only applicable in 3D.

$$\mathbf{A}_s = \begin{pmatrix} \sum_s \mathbf{K}_s^{-1} & \sum_s \mathbf{K}_s^{-1}\mathbf{r}_s^{*T} \\ \sum_s \mathbf{r}_s^*\mathbf{K}_s^{-1} & \sum_s \mathbf{r}_s^*\mathbf{K}_s^{-1}\mathbf{r}_s^{*T} \end{pmatrix}$$

Substituting in $\mathbf{r}_s = \mathbf{c} + \alpha_s\mathbf{u}$

$$\mathbf{A}_s = \begin{pmatrix} \sum_s \mathbf{K}_s^{-1} & \sum_s \mathbf{K}_s^{-1}(\mathbf{c}^* + \alpha_s\mathbf{u}^\star)^T \\ \sum_s (\mathbf{c}^* + \alpha_s\mathbf{u}^\star)\mathbf{K}_s^{-1} & \sum_s (\mathbf{c}^* + \alpha_s\mathbf{u}^\star)\mathbf{K}_s^{-1}(\mathbf{c}^* + \alpha_s\mathbf{u}^\star)^T \end{pmatrix}$$

Expanding this out with the definitions

$$\mathbf{L} = \sum_s \mathbf{K}_s^{-1}, \qquad \mathbf{P} = \sum_s \alpha_s\mathbf{K}_s^{-1}, \qquad \mathbf{Q} = \sum_s \alpha_s^2\mathbf{K}_s^{-1}$$

gives

$$\mathbf{A}_s = \begin{pmatrix} \mathbf{L} & \mathbf{L}\mathbf{c}^{*T} + \mathbf{P}\mathbf{u}^{\star T} \\ \mathbf{c}^*\mathbf{L} + \mathbf{u}^\star\mathbf{P} & \mathbf{c}^*\mathbf{L}\mathbf{c}^{*T} + \mathbf{c}^*\mathbf{P}\mathbf{u}^{\star T} + \mathbf{u}^\star\mathbf{P}\mathbf{c}^{*T} + \mathbf{u}^\star\mathbf{Q}\mathbf{u}^{\star T} \end{pmatrix}.$$

Computing the nullspace of this is aided by the observation that this factors very conveniently as

$$\mathbf{A}_s = \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{c}^* & \boldsymbol{\delta} \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{0} & \mathbf{u}^\star \end{pmatrix} \begin{pmatrix} \mathbf{L} & \mathbf{P} \\ \mathbf{P} & \mathbf{Q} \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{0} & \mathbf{u}^\star \end{pmatrix}^T \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{c}^* & \boldsymbol{\delta} \end{pmatrix}^T. \tag{B.6}$$

The first and last factors are always invertible and thus do not contribute to the nullspace. The second and fourth terms, however, are singular, since the matrix 3×3

matrix $\mathbf{u}^\star$ will always have rank two. This leads to the nullspace (in 3D)

$$\mathbf{Z} = \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{c}^* & \boldsymbol{\delta} \end{pmatrix}^{-T} \begin{pmatrix} \mathbf{0} \\ \mathbf{u} \end{pmatrix} = \begin{pmatrix} \mathbf{c}^*\mathbf{u} \\ \mathbf{u} \end{pmatrix}.$$

We claim that the nullspace of (B.6) has dimension one. First, we must demonstrate that the middle factor of (B.6) is symmetric and positive definite. Note that

$$\begin{aligned} \begin{pmatrix} \mathbf{L} & \mathbf{P} \\ \mathbf{P} & \mathbf{Q} \end{pmatrix} &= \begin{pmatrix} \sum_s \mathbf{K}_s^{-1} & \sum_s \alpha_s \mathbf{K}_s^{-1} \\ \sum_s \alpha_s \mathbf{K}_s^{-1} & \sum_s \alpha_s^2 \mathbf{K}_s^{-1} \end{pmatrix} \\ &= \sum_s \begin{pmatrix} \mathbf{K}_s^{-1} & \alpha_s \mathbf{K}_s^{-1} \\ \alpha_s \mathbf{K}_s^{-1} & \alpha_s^2 \mathbf{K}_s^{-1} \end{pmatrix} \\ &= \sum_s \begin{pmatrix} \boldsymbol{\delta} \\ \alpha_s \boldsymbol{\delta} \end{pmatrix} \mathbf{K}_s^{-1} \begin{pmatrix} \boldsymbol{\delta} \\ \alpha_s \boldsymbol{\delta} \end{pmatrix}^T \end{aligned}$$

Since $\mathbf{K}_s^{-1}$ is SPD, the middle factor of (B.6) is the summation of symmetric positive semidefinite matrices and thus symmetric positive semidefinite itself. It remains only to show that it is nonsingular. A vector lies in the nullspace of a sum of symmetric positive semidefinite matrices if and only if it lies in the nullspace of each summand. Consider one of the summands

$$\begin{pmatrix} \mathbf{K}_s^{-1} & \alpha_s \mathbf{K}_s^{-1} \\ \alpha_s \mathbf{K}_s^{-1} & \alpha_s^2 \mathbf{K}_s^{-1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\delta} \\ \alpha_s \boldsymbol{\delta} \end{pmatrix} \mathbf{K}_s^{-1} \begin{pmatrix} \boldsymbol{\delta} \\ \alpha_s \boldsymbol{\delta} \end{pmatrix}^T.$$

Since the first matrix contains the 3×3 submatrix $\mathbf{K}_s^{-1}$ with full rank, it has rank at least three. The factorization shows that the rank cannot exceed three, so that the matrix has rank exactly three. The vectors of this three dimensional nullspace take the form

$$\begin{pmatrix} \alpha_s \mathbf{w} \\ -\mathbf{w} \end{pmatrix}$$

for any vector $\mathbf{w}$. Consider another summand (using t as the index to distinguish it

from s) multiplied by this vector

$$\begin{pmatrix} \boldsymbol{\delta} \\ \alpha_t \boldsymbol{\delta} \end{pmatrix} \mathbf{K}_s^{-1} \begin{pmatrix} \boldsymbol{\delta} \\ \alpha_t \boldsymbol{\delta} \end{pmatrix}^T \begin{pmatrix} \alpha_s \mathbf{w} \\ -\mathbf{w} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\delta} \\ \alpha_t \boldsymbol{\delta} \end{pmatrix} \mathbf{K}_s^{-1} (\alpha_s \mathbf{w} - \alpha_t \mathbf{w}) = (\alpha_s - \alpha_t) \begin{pmatrix} \mathbf{K}_s^{-1} \mathbf{w} \\ \alpha_t \mathbf{K}_s^{-1} \mathbf{w} \end{pmatrix}.$$

Since $\mathbf{K}_s^{-1}$ is SPD, $\mathbf{K}_s^{-1}\mathbf{w} \neq \mathbf{0}$. Thus, the resulting vector can be zero only if $\alpha_s = \alpha_t$. In particular, this must hold for all choices t. Since $\mathbf{r}_s = \mathbf{c} + \alpha_s \mathbf{u}$, this implies that the intersection points are all coincident, which is not the case being considered. Thus, the middle factor of (B.6) is SPD. Finally it suffices to show that if $\mathbf{B}$ is an SPD matrix and $\mathbf{A}$ is square, then $\mathbf{D} = \mathbf{A}\mathbf{B}\mathbf{A}^T$ has the same rank as $\mathbf{A}$. Indeed, let $\mathbf{w}$ be a nullspace vector of $\mathbf{A}^T$, so that $\mathbf{A}^T\mathbf{w} = \mathbf{0}$. Then, $\mathbf{D}\mathbf{w} = \mathbf{A}\mathbf{B}\mathbf{A}^T\mathbf{w} = \mathbf{0}$. Thus, the rank of $\mathbf{D}$ is no greater than the rank of $\mathbf{A}$. Consider now that $\mathbf{w}$ is a nullspace vector of $\mathbf{D}$. Then, $0 = \mathbf{w}^T\mathbf{D}\mathbf{w} = \mathbf{w}^T\mathbf{A}\mathbf{B}\mathbf{A}^T\mathbf{w} = (\mathbf{A}^T\mathbf{w})\mathbf{B}(\mathbf{A}^T\mathbf{w})$. Since $\mathbf{B}$ is SPD, this implies $\mathbf{A}^T\mathbf{w} = \mathbf{0}$, so that the rank of $\mathbf{D}$ must be exactly the rank of $\mathbf{A}$. Finally, $\mathbf{A}_s$ has exactly one nullspace direction, and this direction is $\mathbf{Z}$. The matrix $\mathbf{Y}$ can be assembled by enumerating vectors orthogonal to $\mathbf{Z}$, using the remaining eigenvectors already computed to provide vectors orthogonal to $\mathbf{u}$.

In the case of all collision points being coincident, $\alpha_s = 0$, and then $\mathbf{P} = \mathbf{Q} = \mathbf{0}$. This simplifies (B.6) to

$$\mathbf{A}_s = \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{c}^* & \boldsymbol{\delta} \end{pmatrix} \begin{pmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{c}^* & \boldsymbol{\delta} \end{pmatrix}^T.$$

From this we obtain the nullspace (in 3D)

$$\mathbf{Z} = \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{c}^* & \boldsymbol{\delta} \end{pmatrix}^{-T} \begin{pmatrix} \mathbf{0} \\ \boldsymbol{\delta} \end{pmatrix} = \begin{pmatrix} -\mathbf{c}^{*T} \\ \boldsymbol{\delta} \end{pmatrix}$$

This corresponds to $\mathbf{A}_s$ having rank three, which is clear from (B.3.3). In this case, the complement is very easy to write down. It is just

$$\mathbf{Y} = \begin{pmatrix} \boldsymbol{\delta} \\ \mathbf{c}^* \end{pmatrix}.$$

In 2D, the outer matrices are not square and thus not invertible. However, we can observe that the same solutions $\mathbf{Z}$ and $\mathbf{Y}$ apply. Indeed, $\mathbf{Y}^T\mathbf{Z} = \mathbf{0}$, so that they are complementary. $\mathbf{Z}$ is 3×1 and $\mathbf{Y}$ is 3×2 (recall that $\mathbf{c}^*$ is 1×2). Further,

$$\mathbf{A}_s\mathbf{Z} = \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{c}^* & \boldsymbol{\delta} \end{pmatrix} \begin{pmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{c}^* & \boldsymbol{\delta} \end{pmatrix}^T \begin{pmatrix} -\mathbf{c}^{*T} \\ \boldsymbol{\delta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{c}^* & \boldsymbol{\delta} \end{pmatrix} \begin{pmatrix} \mathbf{L} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ \boldsymbol{\delta} \end{pmatrix} = \mathbf{0}$$

so that $\mathbf{Z}$ is indeed the nullspace. Since

$$\mathbf{A}_s = \begin{pmatrix} \mathbf{L} & \mathbf{L}\mathbf{c}^{*T} \\ \mathbf{c}^*\mathbf{L} & \mathbf{c}^*\mathbf{L}\mathbf{c}^{*T} \end{pmatrix}.$$

and $\mathbf{L}$ is a nonsingular 2×2 submatrix of $\mathbf{A}_s$, $\mathbf{A}_s$ has rank at least two. Then, $\mathbf{Z}$ is the entire nullspace so that $\mathbf{Y}$ must be the orthogonal complement to the nullspace as desired.

The case of no static outer bodies is equivalent to $\mathbf{Z} = \boldsymbol{\delta}$ and $\mathbf{Y} = \mathbf{0}$. The case where there are more than two bodies and their collision points are not collinear is just $\mathbf{Z} = \mathbf{0}$ and $\mathbf{Y} = \boldsymbol{\delta}$.

Next, consider the right hand side corresponding to the static bodies

$$\mathbf{b}_s = \begin{pmatrix} \sum_s \mathbf{K}_s^{-1}\mathbf{v}_s \\ \sum_s \mathbf{r}_s^*\mathbf{K}_s^{-1}\mathbf{v}_s \end{pmatrix}.$$

In the collinear case this becomes

$$\mathbf{b}_s = \begin{pmatrix} \sum_s \mathbf{K}_s^{-1}\mathbf{v}_s \\ \sum_s \mathbf{c}^*\mathbf{K}_s^{-1}\mathbf{v}_s + \sum_s \alpha_s \mathbf{u}^\star\mathbf{K}_s^{-1}\mathbf{v}_s \end{pmatrix}.$$

This factors as

$$\mathbf{b}_s = \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{c}^* & \boldsymbol{\delta} \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{0} & \mathbf{u}^\star \end{pmatrix} \begin{pmatrix} \sum_s \mathbf{K}_s^{-1}\mathbf{v}_s \\ \sum_s \alpha_s\mathbf{K}_s^{-1}\mathbf{v}_s \end{pmatrix}.$$

From this factorization it is evident that $\mathbf{Z}^T\mathbf{b}_s = 0$.

The result is simplified for the coincident case

$$\mathbf{b}_s = \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{c}^* & \boldsymbol{\delta} \end{pmatrix} \begin{pmatrix} \sum_s \mathbf{K}_s^{-1}\mathbf{v}_s \\ \mathbf{0} \end{pmatrix}$$

This in turn leads to a simpler system for the coincident case

$$\mathbf{A}_s = \begin{pmatrix} \mathbf{L} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\delta} & \mathbf{c}^{*T} \\ \mathbf{0} & \boldsymbol{\delta} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{v} \\ \Delta\boldsymbol{\omega} \end{pmatrix} = \begin{pmatrix} \sum_s \mathbf{K}_s^{-1}\mathbf{v}_s \end{pmatrix}$$

From which it is also evident that $\mathbf{Z}^T\mathbf{b}_s = 0$.

Appendix C

Global Articulation

In this appendix we consider a globally alternative to the pairwise pre-stabilization and post-stabilization of [169]. Post-stabilization is the process by which the components of relative velocity that violate joint constraints are projected out. The operation has very nice properties, as it is a mass-symmetric projection matrix. Indeed, all physically meaningful projections are mass-symmetric. Indeed, if this were not the case, then the projection would be capable of increasing the kinetic energy of the system. Pre-stabilization is a process that is used to correct small errors that accumulate as joints drift apart over time. This process is nonlinear and has a tendency to be tricky to solve well in practice.

If we formulated our articulated rigid bodies based on a reduced set of degrees of freedom, both processes could be avoided most of the time. They would still be required when joint loops are present, though in this case they would only need to be applied to a few joints. If no joint loops exist, then neither process is required. A reduced coordinate representation has the unfortunate side effect of making the representation very complicated. In the maximal coordinate approach taken, all rigid bodies have a uniform representation, which makes processes like collisions relatively simple.

There are two motivations for changing the way in which these problems are solved. The first potential benefit is that the implicit Gauss Seidel solver can be replaced with a solver of choice, which leaves open the possibility of choosing a solver that has better convergence properties. The second reason is one of quality. Solving a global system using an algorithm like conjugate gradient will not show any particular biasing among the joints, so that errors that cannot be resolved will tend to be spread out. By contrast, the pairwise approach tends to be biased by the order in which bodies are treated, and iterative pre-stabilization has poor failure properties. In the case that errors cannot be resolved, most of the error will be concentrated at a few joints. Further, when no solution can be found each iteration continues to make a significant change, leading to additional accumulating errors in the simulation. Note that if the pairwise post-stabilization algorithm is iterated to convergence no biasing will result since it always converges to the unique solution.

C.1 Global Post-stabilization

The globally coupled post-stabilization algorithm is constructed in [168], though it is formulated in a way that may require forming a dense matrix. Further, the solution methods suggested are direct methods, even though the system will often be sparse. The formulation below constructs the system in a way that is factored. This is advantageous since the factors are always sparse, even when the final result would be dense. This factored form is particularly efficient for procedural application of the linear system. It is also convenient since the factors each occur in two different places and can be reused. The global post-stabilization formulation is directly applicable to its intended application in [168]. We also use global post-stabilization as a building block to solve pre-stabilization globally and efficiently.

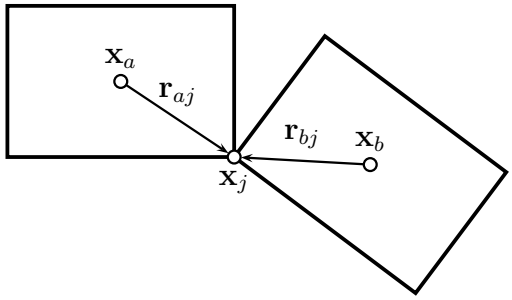


Figure C.1: Two rigid bodies with centers of mass $\mathbf{x}_a$ and $\mathbf{x}_b$ are connected by a joint at $\mathbf{x}_j$.

C.1.1 One Joint

First, we consider the case with two rigid bodies a and b connected by a joint j as shown in Figure C.1. The joint located at $\mathbf{x}_j$, and the centers of mass of the two rigid bodies are at $\mathbf{x}_a$ and $\mathbf{x}_b$. The vectors indicating the displacement of the joint from the centers of mass are $\mathbf{r}_{aj} = \mathbf{x}_j - \mathbf{x}_a$ and $\mathbf{r}_{bj} = \mathbf{x}_j - \mathbf{x}_b$. One of the rigid bodies is taken to be the parent body, and the other body is taken to be the child body. When impulses are applied, the parent body receives the impulse and the child receives the equal and opposite impulse. Similarly, when a relative velocity is computed, the velocity of the child body is subtracted from that of the parent. This distinction is assisted by letting $s_{aj} = 1$ if a is the parent body of joint j, $s_{aj} = -1$ if a is the child body of joint j, and $s_{aj} = 0$ if body a is not connected to joint j (this convention will be convenient later when considering more than two rigid bodies).

Let

$$\mathbf{j}_j = \begin{pmatrix} \hat{\mathbf{j}}_j \\ \hat{\mathbf{j}}_{\boldsymbol{\tau} j} \end{pmatrix} \qquad \mathbf{p}_a = \begin{pmatrix} \hat{\mathbf{p}}_a \\ \hat{\mathbf{L}}_a \end{pmatrix} \qquad \mathbf{v}_a = \begin{pmatrix} \hat{\mathbf{v}}_a \\ \hat{\boldsymbol{\omega}}_a \end{pmatrix}$$

be the generalized impulse (impulse and torque) applied by joint j, generalized momentum (momentum and angular momentum) of body a, and generalized velocity (velocity and angular velocity) of body a. The impulse from joint j experienced by body a is then $s_{aj}\mathbf{j}_j$. The change in angular momentum of body a due to this impulse

is

$$\Delta\mathbf{p}_a = \begin{pmatrix} \Delta\hat{\mathbf{p}}_a \\ \Delta\hat{\mathbf{L}}_a \end{pmatrix} = \begin{pmatrix} s_{aj}\hat{\mathbf{j}}_j \\ s_{aj}\hat{\mathbf{j}}_{\boldsymbol{\tau}a} + \mathbf{r}_{aj} \times s_{aj}\hat{\mathbf{j}}_j \end{pmatrix} = s_{aj} \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{r}^*_{aj} & \boldsymbol{\delta} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{j}}_j \\ \hat{\mathbf{j}}_{\boldsymbol{\tau}j} \end{pmatrix} = \hat{\mathbf{G}}_{aj}\mathbf{j}_j$$

where

$$\hat{\mathbf{G}}_{aj} = s_{aj} \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{r}^*_{aj} & \boldsymbol{\delta} \end{pmatrix}$$

is that matrix that *gathers* joint impulses to rigid bodies. The velocity change produced by this momentum change is

$$\Delta\mathbf{v}_a = \begin{pmatrix} \Delta\hat{\mathbf{v}}_a \\ \Delta\hat{\boldsymbol{\omega}}_a \end{pmatrix} = \begin{pmatrix} m_a^{-1}\Delta\hat{\mathbf{p}}_a \\ \mathbf{I}_a^{-1}\Delta\hat{\mathbf{L}}_a \end{pmatrix} = \mathbf{M}_a^{-1}\Delta\mathbf{p}_a \qquad \mathbf{M}_a = \begin{pmatrix} m_a\boldsymbol{\delta} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_a \end{pmatrix}.$$

Finally, this produces a change in the relative body velocity at the joint location

$$\Delta\mathbf{v}_j = \sum_a s_{aj} \begin{pmatrix} \Delta\hat{\mathbf{v}}_a + \Delta\hat{\boldsymbol{\omega}}_a \times \mathbf{r}_{aj} \\ \Delta\hat{\boldsymbol{\omega}}_a \end{pmatrix} = \sum_a s_{aj} \begin{pmatrix} \boldsymbol{\delta} & \mathbf{r}^{*T}_{aj} \\ \mathbf{0} & \boldsymbol{\delta} \end{pmatrix} \begin{pmatrix} \Delta\hat{\mathbf{v}}_a \\ \Delta\hat{\boldsymbol{\omega}}_a \end{pmatrix} = \sum_a \hat{\mathbf{G}}^T_{aj}\Delta\mathbf{v}_a.$$

The operator $\hat{\mathbf{G}}^T_{aj}$ then *scatters* the rigid body velocity changes to joints. Note the transpose relationship between the impulse gather and velocity scatter. This is a recurring theme in this thesis and in mechanics more generally. Assembling these steps together results in

$$\Delta\mathbf{v}_j = (\hat{\mathbf{G}}^T_{aj}\mathbf{M}_a^{-1}\hat{\mathbf{G}}_{aj} + \hat{\mathbf{G}}^T_{bj}\mathbf{M}_b^{-1}\hat{\mathbf{G}}_{bj})\mathbf{j}_j$$

being the relative velocity change at the location of joint j resulting from an impulse $\mathbf{j}_j$ applied at that location. This is the same form that was obtained in [168]. This equation shows how to choose an impulse such that a desired relative joint velocity is obtained. For a fully constrained joint (two rigid bodies fused together), it should be the case that

$$\mathbf{v}_j + \Delta\mathbf{v}_j = \sum_a \hat{\mathbf{G}}^T_{aj}(\mathbf{v}_a + \Delta\mathbf{v}_a) = \mathbf{0},$$

where $\mathbf{v}_j$ is the relative velocity of the parent and child bodies at the location of joint

j. That is, the desired relative velocity is zero. This statement is not true for more general joints, since relative motion in some directions is unconstrained. The velocity constraint leads to

$$\begin{aligned}
\sum_a \hat{\mathbf{G}}_{aj}^T(\mathbf{v}_a + \Delta\mathbf{v}_a) &= \mathbf{0} \\
\sum_a \hat{\mathbf{G}}_{aj}^T \Delta\mathbf{v}_a &= -\sum_a \hat{\mathbf{G}}_{aj}^T \mathbf{v}_a \\
\sum_a \hat{\mathbf{G}}_{aj}^T \mathbf{M}_a^{-1} \hat{\mathbf{G}}_{aj} \mathbf{j}_j &= -\sum_a \hat{\mathbf{G}}_{aj}^T \mathbf{v}_a \\
(\hat{\mathbf{G}}_{aj}^T \mathbf{M}_a^{-1} \hat{\mathbf{G}}_{aj} + \hat{\mathbf{G}}_{bj}^T \mathbf{M}_b^{-1} \hat{\mathbf{G}}_{bj}) \mathbf{j}_j &= -\hat{\mathbf{G}}_{aj}^T \mathbf{v}_a - \hat{\mathbf{G}}_{bj}^T \mathbf{v}_b
\end{aligned}$$

The final body velocities are obtained as

$$\mathbf{v}_a^{n+1} = \mathbf{v}_a + \Delta\mathbf{v}_a = \mathbf{v}_a + \mathbf{M}_a^{-1} \hat{\mathbf{G}}_{aj} \mathbf{j}_j.$$

Next, we turn to the velocity change that is desired. Let $\mathbf{P}_j$ be the projection that removes degrees of freedom that the joint j does not constrain. For a rigid joint, for instance, $\mathbf{P}_j = \boldsymbol{\delta}$. For a joint that permits arbitrary rotation but prevents translation, $\mathbf{P}_j = \begin{pmatrix} \boldsymbol{\delta} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$. Note that this is not the convention taken in [168], which defines $\mathbf{P}_j$ such that it removes degrees of freedom rather than simply projecting them out. While it is advantageous to implement this operator by removing degrees of freedom, thus removing the nullspace from the system and making it smaller, we retain the degrees of freedom for convenience. In our case, $\mathbf{P}_j^2 = \mathbf{P}_j = \mathbf{P}_j^T$. Following [168] we project the torque to avoid applying torque in free directions, and we project the relative joint velocity to ignore relative velocity in directions that are unconstrained and should therefore not be eliminated by the impulse. The new system and update rule are

$$\sum_a \mathbf{P}_j \hat{\mathbf{G}}_{aj}^T \mathbf{M}_a^{-1} \hat{\mathbf{G}}_{aj} \mathbf{P}_j \mathbf{j}_j = -\sum_a \mathbf{P}_j \hat{\mathbf{G}}_{aj}^T \mathbf{v}_a \qquad \mathbf{v}_a^{n+1} = \mathbf{v}_a + \mathbf{M}_a^{-1} \hat{\mathbf{G}}_{aj} \mathbf{P}_j \mathbf{j}_j$$

Finally, let $\mathbf{G}_{aj} = \hat{\mathbf{G}}_{aj}\mathbf{P}_j$ for convenience. Then, system and update rule are

$$\sum_a \mathbf{G}_{aj}^T \mathbf{M}_a^{-1} \mathbf{G}_{aj} \mathbf{j}_j = -\sum_a \mathbf{G}_{aj}^T \mathbf{v}_a \qquad \mathbf{v}_a^{n+1} = \mathbf{v}_a + \mathbf{M}_a^{-1} \mathbf{G}_{aj} \mathbf{j}_j.$$

C.1.2 Linear System

For multiple joints and bodies, (C.1) must be changed slightly. Rigid bodies move in response to the net torque applied to them, so that

$$\Delta \mathbf{p}_a = \sum_j \hat{\mathbf{G}}_{aj} \mathbf{j}_j.$$

Then, (C.1) becomes

$$\sum_a \mathbf{G}_{aj}^T \mathbf{M}_a^{-1} \sum_g \mathbf{G}_{ag} \mathbf{j}_g = -\sum_a \mathbf{G}_{aj}^T \mathbf{v}_a \qquad \mathbf{v}_a^{n+1} = \mathbf{v}_a + \mathbf{M}_a^{-1} \mathbf{G}_{aj} \mathbf{j}_j,$$

where f and g both represent joints.

Next we extend this to a full system of rigid bodies and joints. Let $\mathbf{v}$ be the vector of generalized velocities with an entry $\mathbf{v}_a$ for each body a. Let $\mathbf{j}$ be the vector of generalized impulses with one entry $\mathbf{j}_j$ for each joint j. Let $\mathbf{G}$ be the block matrix whose blocks are $\mathbf{G}_{aj}$. Finally, let $\mathbf{M}$ be the block diagonal matrix whose diagonal blocks are given by $\mathbf{M}_a$. (C.1) now takes the elegant form

$$\mathbf{G}^T \mathbf{M}^{-1} \mathbf{G} \mathbf{j} = -\mathbf{G}^T \mathbf{v} \qquad \mathbf{v}^{n+1} = \mathbf{v} + \mathbf{M}^{-1} \mathbf{G} \mathbf{j}.$$

Note that this is almost identical to the pressure projection for fluids, which took the form

$$\mathbf{G}^T \mathbf{M}^{-1} \mathbf{G} \hat{\mathbf{p}} = \mathbf{G}^T \mathbf{u}^* \qquad \mathbf{u}^{n+1} = \mathbf{u}^* - \mathbf{M}^{-1} \mathbf{G} \hat{\mathbf{p}}.$$

Besides the use of $\mathbf{G}$ for the gather matrix, this similarity is not coincidental. Both systems represent a velocity projection to satisfy a linear velocity constraint.

C.1.3 Early Termination

The similarity of the post-stabilization system and the fluids pressure system is convenient for another reason. The pressure system can be formulated as choosing impulses $\mathbf{j}$ to minimize the kinetic energy of the system (See [16] and Section 5.9). That is,

$$\begin{aligned} KE &= \frac{1}{2}((\mathbf{v}^{n+1})^T\mathbf{M}\mathbf{v}^{n+1} \\ &= \frac{1}{2}(\mathbf{v}+\mathbf{M}^{-1}\mathbf{G}\mathbf{j})^T\mathbf{M}(\mathbf{v}+\mathbf{M}^{-1}\mathbf{G}\mathbf{j}) \\ &= \frac{1}{2}\mathbf{v}^T\mathbf{M}\mathbf{v}+\mathbf{v}^T\mathbf{G}\mathbf{j}+\frac{1}{2}\mathbf{j}^T\mathbf{G}^T\mathbf{M}^{-1}\mathbf{G}\mathbf{j} \\ \mathbf{0} &= \frac{\partial KE}{\partial \mathbf{j}} \\ &= \frac{\partial}{\partial \mathbf{j}}\left(\frac{1}{2}\mathbf{v}^T\mathbf{M}\mathbf{v}+\mathbf{v}^T\mathbf{G}\mathbf{j}+\frac{1}{2}\mathbf{j}^T\mathbf{G}^T\mathbf{M}^{-1}\mathbf{G}\mathbf{j}\right) \\ &= \mathbf{G}^T\mathbf{v}+\mathbf{G}^T\mathbf{M}^{-1}\mathbf{G}\mathbf{j} \\ \mathbf{G}^T\mathbf{M}^{-1}\mathbf{G}\mathbf{j} &= -\mathbf{G}^T\mathbf{v} \end{aligned}$$

This is just (C.1). The conjugate gradient solver has the property that chooses its iterates within each Krylov subspace to minimize the quantity $\mathbf{G}^T\mathbf{v}+\mathbf{G}^T\mathbf{M}^{-1}\mathbf{G}\mathbf{j}$ (and thus the kinetic energy, since $\frac{1}{2}\mathbf{v}^T\mathbf{M}\mathbf{v}$ does not depend on $\mathbf{j}$). This is interesting because it tells us that if (unpreconditioned) conjugate gradient is used to solve this system, then the total kinetic energy of the system will not increase, even if the conjugate gradient solver is terminated early.

Unfortunately, this property is unlikely to be retained if preconditioning is used. Indeed, if $\mathbf{S}$ is the preconditioner, a symmetric and positive definite operator typically chosen such that $\mathbf{S}(\mathbf{G}^T\mathbf{M}^{-1}\mathbf{G}) \approx \boldsymbol{\delta}$, then the quantity being minimized at every iteration is

$$\mathbf{S}^{\frac{1}{2}}\mathbf{G}^T\mathbf{v}+\mathbf{S}^{\frac{1}{2}}\mathbf{G}^T\mathbf{M}^{-1}\mathbf{G}\mathbf{S}^{\frac{1}{2}}\mathbf{j}.$$

This is unfortunate, as convergence with conjugate gradient can be quite poor in practice. Note that a similar argument can be made for the pressure solve in a typical Navier-Stokes solver.

C.2 Global Pre-stabilization

Let ϕ be the function that maps from generalized velocities to frames,

$$\phi(\mathbf{v}) = \phi\begin{pmatrix}\hat{\mathbf{v}}\\ \hat{\boldsymbol{\omega}}\end{pmatrix} = \begin{pmatrix}\hat{\mathbf{v}}\\ e^{\hat{\boldsymbol{\omega}}^*}\end{pmatrix}.$$

Note that in last expression, the "vector" really contains a vector $\hat{\mathbf{v}}$ and a matrix $e^{\hat{\boldsymbol{\omega}}^*}$. The block vector notation is only used for convenience. The frame that this "vector" represents is described by its effect on a vector $\mathbf{z}$. In particular, a frame has the affect of transforming a vector $\mathbf{z}$ according to

$$\mathbf{F}\mathbf{z} = \begin{pmatrix}\mathbf{x}\\ \mathbf{R}\end{pmatrix}\mathbf{z} = \mathbf{R}\mathbf{z} + \mathbf{x},$$

where it is assumed that $\mathbf{R}$ is a rotation matrix. Let ψ be a suitable inverse function (ignoring branches) such that $\phi(\psi(\mathbf{F})) = \mathbf{F}$. This function is given by

$$\psi(\mathbf{F}) = \psi\begin{pmatrix}\mathbf{x}\\ \mathbf{R}\end{pmatrix} = \begin{pmatrix}\mathbf{x}\\ [\ln \mathbf{R}]\end{pmatrix},$$

where $[\ln \mathbf{R}]$ is meant to indicate that the vector $\mathbf{u}$ is chosen, where $\mathbf{u}^* = \ln \mathbf{R}$, or alternatively $\mathbf{R} = e^{\mathbf{u}^*}$.

Consider two rigid bodies with frames $\mathbf{F}_a$ and $\mathbf{F}_b$. Ignoring the usage of higher order orientation evolution, the desired frames to which the bodies will evolve are $\mathbf{F}_{a\star} = \phi(\Delta t \mathbf{v}_a)\mathbf{F}_a$ and $\mathbf{F}_{b\star} = \phi(\Delta t \mathbf{v}_b)\mathbf{F}_b$. Let the joint to parent (body a) and child (body b) frames be $\mathbf{F}_p$ and $\mathbf{F}_c$. The frame from joint-parent to joint-child is $\mathbf{F}_j = {\mathbf{F}_c}^{-1}{\mathbf{F}_b}^{-1}\mathbf{F}_a\mathbf{F}_p$ (see [169]). Evolving $\mathbf{F}_a$ and $\mathbf{F}_b$ to $\mathbf{F}_{a\star}$ and $\mathbf{F}_{b\star}$ results in the modified joint frame $\hat{\mathbf{F}}_j = {\mathbf{F}_c}^{-1}{\mathbf{F}_{b\star}}^{-1}\mathbf{F}_{a\star}\mathbf{F}_p$. Since the goal is to fix the joint frames to respect the joint constraints, the frame $\hat{\mathbf{F}}_j$ is corrected by projecting out invalid displacements. Pre-stabilization is supposed to choose impulses to target these new

joint frames. Then,

$$\begin{aligned}
\hat{\mathbf{F}}_j &= {\mathbf{F}_c}^{-1}{\mathbf{F}_{b\star}}^{-1}\mathbf{F}_{a\star}\mathbf{F}_p \\
{\mathbf{F}_{b\star}}^{-1}\mathbf{F}_{a\star} &= \mathbf{F}_c\hat{\mathbf{F}}_j{\mathbf{F}_p}^{-1} \\
\mathbf{F}_c\hat{\mathbf{F}}_j{\mathbf{F}_p}^{-1} &= (\phi(\Delta t\mathbf{v}_b)\mathbf{F}_b)^{-1}\phi(\Delta t\mathbf{v}_a)\mathbf{F}_a \\
\mathbf{F}_b\mathbf{F}_c\hat{\mathbf{F}}_j{\mathbf{F}_p}^{-1}{\mathbf{F}_a}^{-1} &= \phi(\Delta t\mathbf{v}_b)^{-1}\phi(\Delta t\mathbf{v}_a) \\
&\approx \phi(\Delta t\mathbf{v}_a - \Delta t\mathbf{v}_b) \\
\mathbf{v}_{rel} &\approx \Delta t^{-1}\psi(\mathbf{F}_b\mathbf{F}_c\hat{\mathbf{F}}_j{\mathbf{F}_p}^{-1}{\mathbf{F}_a}^{-1})
\end{aligned}$$

This is the desired relative joint velocity for one joint. These form the right hand side of a coupled solve for pre-stabilization. The system is otherwise formed as described in Section C.1.2. This still leaves the question of whether the projections $\mathbf{P}$ should be included in pre-stabilization. Since impulses along free directions, though small, will likely cause longterm noticeable effects, the projections should probably be included $\mathbf{P}$ as they are for pre-stabilization. Note that the pairwise algorithm of [169] does not consider the possibility of projecting during pre-stabilization in this way.

There are two sources of error in this pre-stabilization formulation. The first is in the assumption of first order evolution rather than the second order evolution that will actually be used. This approximation is reasonable since it is a second order error, and there is a second source of error anyway. This second error comes from the approximation $\phi(\Delta t\mathbf{v}_b)^{-1}\phi(\Delta t\mathbf{v}_a) \approx \phi(\Delta t\mathbf{v}_a - \Delta t\mathbf{v}_b)$. It remains to show that this error is also second order. First, it is useful to recall two algebraic properties of frames, which can be derived by noting their operation on a vector $\mathbf{z}$

$$\begin{pmatrix}\mathbf{x}_a\\ \mathbf{R}_a\end{pmatrix}\begin{pmatrix}\mathbf{x}_b\\ \mathbf{R}_b\end{pmatrix} = \begin{pmatrix}\mathbf{x}_a + \mathbf{R}_a\mathbf{x}_b\\ \mathbf{R}_a\mathbf{R}_b\end{pmatrix} \qquad \begin{pmatrix}\mathbf{x}\\ \mathbf{R}\end{pmatrix}^{-1} = \begin{pmatrix}-\mathbf{R}^{-1}\mathbf{x}\\ \mathbf{R}^{-1}\end{pmatrix}.$$

Then,

$$\begin{pmatrix}\mathbf{x}_a\\ \mathbf{R}_a\end{pmatrix}^{-1}\begin{pmatrix}\mathbf{x}_b\\ \mathbf{R}_b\end{pmatrix} = \begin{pmatrix}-\mathbf{R}_a^{-1}\mathbf{x}_a\\ \mathbf{R}_a^{-1}\end{pmatrix}\begin{pmatrix}\mathbf{x}_b\\ \mathbf{R}_b\end{pmatrix} = \begin{pmatrix}-\mathbf{R}_a^{-1}\mathbf{x}_a + \mathbf{R}_a^{-1}\mathbf{x}_b\\ \mathbf{R}_a^{-1}\mathbf{R}_b\end{pmatrix} = \begin{pmatrix}\mathbf{R}_a^{-1}(-\mathbf{x}_a + \mathbf{x}_b)\\ \mathbf{R}_a^{-1}\mathbf{R}_b\end{pmatrix}$$

Next, we compute a Taylor expansion of $\phi(\Delta t\mathbf{v}_a - \Delta t\mathbf{v}_b)$.

$$\begin{aligned}
\phi(\Delta t\mathbf{v}_a - \Delta t\mathbf{v}_b) &= \phi\begin{pmatrix}\Delta t\hat{\mathbf{v}}_a - \Delta t\hat{\mathbf{v}}_b \\ \Delta t\hat{\boldsymbol{\omega}}_a - \Delta t\hat{\boldsymbol{\omega}}_b\end{pmatrix} \\
&= \begin{pmatrix}\Delta t\hat{\mathbf{v}}_a - \Delta t\hat{\mathbf{v}}_b \\ e^{\Delta t\hat{\boldsymbol{\omega}}_a^* - \Delta t\hat{\boldsymbol{\omega}}_b^*}\end{pmatrix} \\
&\approx \begin{pmatrix}\Delta t\hat{\mathbf{v}}_a - \Delta t\hat{\mathbf{v}}_b \\ \boldsymbol{\delta} + \Delta t(\hat{\boldsymbol{\omega}}_a^* - \hat{\boldsymbol{\omega}}_b^*)\end{pmatrix}
\end{aligned}$$

This is followed by the Taylor expansion of $\phi(\Delta t\mathbf{v}_b)^{-1}\phi(\Delta t\mathbf{v}_a)$.

$$\begin{aligned}
\phi(\Delta t\mathbf{v}_b)^{-1}\phi(\Delta t\mathbf{v}_a) &= \phi\begin{pmatrix}\Delta t\hat{\mathbf{v}}_b \\ \Delta t\hat{\boldsymbol{\omega}}_b\end{pmatrix}^{-1}\phi\begin{pmatrix}\Delta t\hat{\mathbf{v}}_a \\ \Delta t\hat{\boldsymbol{\omega}}_a\end{pmatrix} \\
&= \begin{pmatrix}\Delta t\hat{\mathbf{v}}_b \\ e^{\Delta t\hat{\boldsymbol{\omega}}_b^*}\end{pmatrix}^{-1}\begin{pmatrix}\Delta t\hat{\mathbf{v}}_a \\ e^{\Delta t\hat{\boldsymbol{\omega}}_a^*}\end{pmatrix} \\
&= \begin{pmatrix}e^{-\Delta t\hat{\boldsymbol{\omega}}_b^*}(\Delta t\hat{\mathbf{v}}_a - \Delta t\hat{\mathbf{v}}_b) \\ e^{-\Delta t\hat{\boldsymbol{\omega}}_b^*}e^{\Delta t\hat{\boldsymbol{\omega}}_a^*}\end{pmatrix} \\
&\approx \begin{pmatrix}\Delta t\hat{\mathbf{v}}_a - \Delta t\hat{\mathbf{v}}_b \\ \boldsymbol{\delta} + \Delta t(\hat{\boldsymbol{\omega}}_a^* - \hat{\boldsymbol{\omega}}_b^*)\end{pmatrix}
\end{aligned}$$

where second and higher order terms have been dropped. We see that the two expansions agree to first order.

The resulting global pre-stabilization scheme produces only a first order correction to the positional error and must be iterated to convergence. In practice, a few iterations of this procedure typically suffice, since it tends to converge rapidly. Note that unlike the pairwise approach, this global approach does not utilize limiters and may therefore diverge. Though it may (and occasionally will) diverge if very large errors are present, the method seems to be fairly robust in practice, even when large numbers of bodies, loops, and joints are present and the rigid bodies are started in a scrambled configuration.

Bibliography

[1] A. Alizadeh Pahlavan and J.B. Freund. Effect of solid properties on slip at a fluid-solid interface. Phys. Rev. E, 83(2), 2011.

[2] A. Angelidis, M.P. Cani, G. Wyvill, and S. King. Swirling-sweepers: constant volume modeling. Graph. Models, 68(4):324–32, 2006.

[3] F. Armero and E Petocz. Formulation and analysis of conserving algorithms for frictionless dynamic contact/impact problems. Computer methods in applied mechanics and engineering, 158(3):269–300, 1998.

[4] F.P.T. Baaijens. A fictitious domain/mortar element method for fluid-structure interaction. Intl. J. for Num. Meth. in Fluids, 35(7):743–761, 2001.

[5] Z. Bao, J.M. Hong, J. Teran, and R. Fedkiw. Fracturing rigid materials. IEEE Trans. Viz. Comput. Graph., 13:370–378, 2007.

[6] D. Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. Comput. Graph. (Proc. SIGGRAPH 89), 23(3):223–232, 1989.

[7] D. Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. Comput. Graph. (Proc. SIGGRAPH 90), 24(4):19–28, 1990.

[8] D. Baraff. Coping with friction for non-penetrating rigid body simulation. Comput. Graph. (Proc. SIGGRAPH 91), 25(4):31–40, 1991.

[9] D. Baraff. Fast contact force computation for nonpenetrating rigid bodies. In Proc. SIGGRAPH 94, pages 23–34, 1994.

[10] D. Baraff. Linear-time dynamics using Lagrange multipliers. In Proc. of ACM SIGGRAPH 1996, pages 137–146, 1996.

[11] D. Baraff and A. Witkin. Partitioned dynamics. Technical report, Carnegie Mellon University, 1997.

[12] D. Baraff and A. Witkin. Large steps in cloth simulation. In ACM SIGGRAPH 98, pages 43–54. ACM Press/ACM SIGGRAPH, 1998.

[13] R. Barzel, J. Hughes, and D. Wood. Plausible motion simulation for computer graphics animation. In Comput. Anim. and Sim. '96, Proc. Eurographics Wrkshp., pages 183–197, 1996.

[14] K.J. Bathe and H. Zhang. Finite element developments for general fluid flows with structural interactions. International Journal for Numerical Methods in Engineering, 60(1), 2004.

[15] C. Batty, F. Bertails, and R. Bridson. A fast variational framework for accurate solid-fluid coupling. ACM Trans. Graph. (SIGGRAPH Proc.), 26(3):100, 2007.

[16] C. Batty, F. Bertails, and R. Bridson. A fast variational framework for accurate solid-fluid coupling. ACM Trans. Graph. (SIGGRAPH Proc.), 26(3), 2007.

[17] J. Bender. Impulse-based dynamic simulation in linear time. Computer Animation and Virtual Worlds, 18:225–233, 2007.

[18] M. Bertalmio, L.T. Cheng, S. Osher, and G. Sapiro. Variational problems and partial differential equations on implicit surfaces: the framework and examples in image processing and pattern formation, 2000.

[19] I. Bijelonja, I. Demirdžić, and S. Muzaferija. A finite volume method for large strain analysis of incompressible hyperelastic materials. Int. J. Num. Meth. Eng., 64:1594–1609, 2005.

[20] I. Bijelonja, I. Demirdžić, and S. Muzaferija. A finite volume method for incompressible linear elasticity. Comput. Meth. Appl. Meth. Eng., 195:6378–90, 2006.

[21] J. Bonet and A. Burton. A simple average nodal pressure tetrahedral element for incompressible and nearly incompressible dynamic explicit applications. Comm. Num. Meth. Eng., 14:437–449, 1998.

[22] J. Bonet and R. Wood. Nonlinear continuum mechanics for finite element analysis. Cambridge University Press, Cambridge, 1997.

[23] B. Boroomand and B. Khalilian. On using linear elements in incompressible plane strain problems: a simple edge based approach for triangles. Int. J. Num. Meth. Eng., 61:1710–1740, 2004.

[24] N. Bou-Rabee and J.E. Marsden. Hamilton-pontryagin integrators on lie groups part i: Introduction and structure-preserving properties. Foundations of Computational Mathematics, 9(2):1615–3375, 2009.

[25] D. Bourguignon and M. P. Cani. Controlling anisotropy in mass-spring systems. In Eurographics, pages 113–123. Eurographics Assoc., 2000.

[26] J. U. Brackbill, D. B. Kothe, and C. Zemach. A continuum method for modelling surface tension. J. Comput. Phys., 100:335–353, 1992.

[27] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. ACM Trans. Graph., 21(3):594–603, 2002.

[28] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim., pages 28–36, 2003.

[29] T.M. Burton and J.K. Eaton. Analysis of a fractional-step method on overset grids. J. of Comput. Phys., 177(2):336–364, 2002.

[30] P. Causin, J.-F. Gerbeau, and F. Nobile. Added-mass effect in the design of partitioned algorithms for fluid-structure problems. Comp. Meth. Appl. Mech. Engng., 194(42-44), 2005.

[31] Y.C. Chang, T.Y. Hou, B. Merriman, and S. Osher. A level set formulation of eulerian interface capturing methods for incompressible fluid flows. J. Comput. Phys., 124(2):449–464, 1996.

[32] A. Chatterjee and A. Ruina. A new algebraic rigid body collision law based on impulse space considerations. ASME J. Appl. Mech., 65(4):939–951, 1998.

[33] J. Chessa and T. Belytschko. An enriched finite element method and level sets for axisymmetric two-phase flow with surface tension. Int. J. Num. Meth. Eng., 58:2041–2064, 2003.

[34] J. Chessa and T. Belytschko. An extended finite element method for two-phase fluids. ASME J. Appl. Mech., 70:10–17, 2003.

[35] S.-C. Choi. Iterative Methods for Singular Linear Equations and Least-Squares Problems. PhD thesis, Stanford University, 2006.

[36] D. Chopp. Some improvements of the fast marching method. SIAM J. Sci. Comput., 223:230–244, 2001.

[37] A. Chorin. A numerical method for solving incompressible viscous flow problems. J. Comput. Phys., 2:12–26, 1967.

[38] B. Cockburn, D. Schötzau, and J. Wang. Discontinuous Galerkin methods for incompressible elastic materials. Comput. Meth. Appl. Meth. Eng., 195:3184–3204, 2006.

[39] L. Cooper and S. Maddock. Preventing collapse within mass-spring-damper models of deformable objects. In 5th Int. Conf. in Central Europe on Comput. Graph. and Vis., 1997.

[40] B. Criswell, K. Derlich, and D. Hatch. Davy jones' beard: rigid tentacle simulation. In SIGGRAPH 2006 Sketches, page 117, 2006.

[41] E. de Souza Neto, F. Andrade Pires, and D. Owen. F-bar-based linear triangles and tetrahedra for finite strain analysis of nearly incompressible solids. Part I: formulation and benchmarking. Int. J. Num. Meth. Eng., 62:353–383, 2005.

[42] M. Desbrun and M.-P. Gascuel. Animating soft substances with implicit surfaces. In Proc. SIGGRAPH 95, pages 287–290, 1995.

[43] J.E. Dolbow and A. Devan. Enrichment of enhanced assumed strain approximation for representing strong discontinuities: Addressing volumetric incompressibility and the discontinuous patch test. Int. J. Num. Meth. Eng., 59:47–67, 2004.

[44] D. Dos Santos, N. Miguel, J.F. Gerbeau, and J.F. Bourgat. A partitioned fluid-structure algorithm for elastic thin valves with contact. Computer Methods in Applied Mechanics and Engineering, 2006.

[45] J. Douglas Jr and T. Dupont. Alternating-direction Galerkin methods on rectangles. Numer. Sol. of Part. Diff. Eqns., 2:133–214.

[46] B. Engquist, A.-K. Tornberg, and R. Tsai. Discretization of Dirac delta functions in level set methods. J. Comput. Phys., 207(1):28–51, 2005.

[47] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. J. Comput. Phys., 183:83–116, 2002.

[48] D. Enright, D. Nguyen, F. Gibou, and R. Fedkiw. Using the particle level set method and a second order accurate pressure boundary condition for free surface flows. In Proc. 4th ASME-JSME Joint Fluids Eng. Conf., number FEDSM2003–45144. ASME, 2003.

[49] P. Esser, J. Grande, and A. Reusken. An extended finite element method applied to levitated droplet problems. Intl. J. for Num. Meth. in Engng., 2010.

[50] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Composable controllers for physics-based character animation. In ACM Trans. Graph. (SIGGRAPH Proc.), pages 251–260, 2001.

[51] C. Farhat, M. Lesoinne, and P. Le Tallec. Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: momentum and energy conservation, optimal discretization and application to aeroelasticity. Computer methods in applied mechanics and engineering, 157(1-2):95–114, 1998.

[52] R. Fedkiw, J. Stam, and H. Jensen. Visual simulation of smoke. In Proc. of ACM SIGGRAPH 2001, pages 15–22, 2001.

[53] B. Feldman, J. O'Brien, and O. Arikan. Animating suspended particle explosions. ACM Trans. Graph. (SIGGRAPH Proc.), 22(3):708–715, 2003.

[54] M.M. Francois, S.J. Cummins, E.D. Dendy, D.B. Kothe, J.M. Sicilian, and M.W. Williams. A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework. J. of Comput. Phys., 213(1):141–173, 2006.

[55] M.M. Francois, D.B. Kothe, and S.J. Cummins. Modelling Surface Tension Using a Ghost Fluid Technique within a Volume of Fluid Formulation. Technical report, Los Alamos National Laboratory, 2004.

[56] N. Galoppo, M. Otaduy, S. Tekin, M. Gross, and M. C. Lin. Soft articulated characters with fast contact handling. Comput. Graph. Forum (Proc. Eurographics), 26(3):243–253, 2007.

[57] F. Gibou, R. Fedkiw, L.-T. Cheng, and M. Kang. A second-order-accurate symmetric discretization of the Poisson equation on irregular domains. J. Comput. Phys., 176:205–227, 2002.

[58] J.P. Gois, A. Nakano, L.G. Nonato, and G.C. Buscaglia. Front tracking with moving-least-squares surfaces. J. of Comput. Phys., 227(22):9643–9669, 2008.

[59] R. Goldenthal, D. Harmon, R. Fattal, M. Bercovier, and E. Grinspun. Efficient simulation of inextensible cloth. ACM Trans. Graph., 26(3):49, 2007.

[60] B.E. Griffith, R.D. Hornung, D.M. McQueen, and C.S. Peskin. An adaptive, formally second order accurate version of the immersed boundary method. J. of Comput. Phys., 223(1):10–49, 2007.

[61] B.E. Griffith and C.S. Peskin. On the order of accuracy of the immersed boundary method: higher order convergence rates for sufficiently smooth problems. J. Comput. Phys., 208(1):75–105, 2005.

[62] S. Gross and A. Reusken. An extended pressure finite element space for two-phase incompressible flows with surface tension. J. of Comput. Phys., 224(1):40–58, 2007.

[63] S. Gross and A. Reusken. Finite Element Discretization Error Analysis of a Surface Tension Force in Two-Phase Incompressible Flows. SIAM Journal on Numerical Analysis, 45(4):1679–1700, 2007.

[64] E. Guendelman, R. Bridson, and R. Fedkiw. Nonconvex rigid bodies with stacking. ACM Trans. Graph. (SIGGRAPH Proc.), 22(3):871–878, 2003.

[65] E. Guendelman, A. Selle, F. Losasso, and R. Fedkiw. Coupling water and smoke to thin deformable and rigid shells. ACM Trans. Graph. (SIGGRAPH Proc.), 24(3):973–981, 2005.

[66] J. Hahn. Realistic animation of rigid bodies. Comput. Graph. (Proc. SIGGRAPH 88), 22(4):299–308, 1988.

[67] J. Happel and H. Brenner. Low Reynolds number hydrodynamics: with special applications to particulate media. Kluwer Academic Print on Demand, 1991.

[68] F. Harlow and J. Welch. Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. Phys. Fluids, 8:2182–2189, 1965.

[69] A. Harten, B. Enquist, S. Osher, and S. Chakravarthy. Uniformly high-order accurate essentially non-oscillatory schemes III. J. Comput. Phys., 71:231–303, 1987.

[70] J.J. Helmsen, E.G. Puckett, P. Colella, and M. Dorr. Two new methods for simulating photolithography development in 3D. In Proceedings of SPIE, volume 2726, page 253, 1996.

[71] W.D. Henshaw. A fourth-order accurate method for the incompressible Navier-Stokes equations on overlapping grids. J. of Comput. Phys., 113(1):13–25, 1994.

[72] J. Hochstein and T. Williams. An implicit surface tension model. AIAA Meeting Papers., 599, 1996.

[73] J. Hodgins, W. Wooten, D. Brogan, and J. O'Brien. Animating human athletics. In Proc. of SIGGRAPH '95, pages 71–78, 1995.

[74] M. Hong, S. Jung, M.H. Choi, and S. Welch. Fast volume preservation for a mass-spring system. IEEE Comput. Graph. and Appl., 26:83–91, September 2006.

[75] T.Y. Hou, J.S. Lowengrub, and M.J. Shelley. Removing the stiffness from interfacial flows with surface tension. J. of Comput. Phys., 114(2):312–338, 1994.

[76] T. J. R. Hughes. The Finite Element Method: Linear Static and Dynamic Finite Element Analysis. Dover, 2000.

[77] S. Hysing. A new implicit surface tension implementation for interfacial flows. International Journal for Numerical Methods in Fluids, 51(6):659–672, 2006.

[78] Hayley N. Iben and James F. O'Brien. Generating surface crack patterns. pages 177–185, 2006.

[79] SR Idelsohn, J. Marti, A. Limache, and E. Onate. Unified lagrangian formulation for elastic solids and incompressible fluids: Application to fluid–structure

interaction problems via the PFEM. Computer Methods in Applied Mechanics and Engineering, 197(19-20):1762–1776, 2008.

[80] G. Irving, C. Schroeder, and R. Fedkiw. Volume conserving finite element simulations of deformable models. ACM Trans. Graph. (SIGGRAPH Proc.), 26(3):13.1–13.6, 2007.

[81] G. Irving, J. Teran, and R. Fedkiw. Invertible finite elements for robust simulation of large deformation. In Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim., pages 131–140, 2004.

[82] A.J. James and J. Lowengrub. A surfactant-conserving volume-of-fluid method for interfacial flows with insoluble surfactant. J. of Comput. Phys., 201(2):685–722, 2004.

[83] D. Jamet, D. Torres, and JU Brackbill. On the theory and computation of surface tension: the elimination of parasitic currents through energy conservation in the second-gradient method. J. of Comput. Phys., 182(1):262–276, 2002.

[84] J. Jansson and J.S.M. Vergeest. Combining deformable and rigid body mechanics simulation. The Vis. Comput. J., 2003.

[85] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of Hermite data. ACM Trans. Graph. (SIGGRAPH Proc.), 21(3):339–346, 2002.

[86] M. Kang, R. Fedkiw, and X.-D. Liu. A boundary condition capturing method for multiphase incompressible flow. J. Sci. Comput., 15:323–360, 2000.

[87] D. Kaufman, T. Edmunds, and D. Pai. Fast frictional dynamics for rigid bodies. ACM Trans. Graph. (SIGGRAPH Proc.), 24(3):946–956, 2005.

[88] D. Kaufman, S. Sueda, D. James, and D. Pai. Staggered projections for frictional contact in multibody systems. ACM Transactions on Graphics (SIGGRAPH Asia 2008), 27(5):164:1–164:11, 2008.

[89] L. Kharevych, W. Yang, Y. Tong, E. Kanso, J. Marsden, P. Schröder, and M. Desbrun. Geometric variational integrators for computer animation. ACM SIGGRAPH/Eurographics Symp. on Comput. Anim., pages 43–51, 2006.

[90] Y. Kim, M.C. Lai, and C.S. Peskin. Numerical simulations of two-dimensional foam by the immersed boundary method. J. of Comput. Phys., 229(13):5194–5207, 2010.

[91] M. Kobilarov, K. Crane, and M. Desbrun. Lie group integrators for animation and control of vehicles. ACM Transactions on Graphics, 28(2), April 2009.

[92] S. Lahiri, J. Bonet, J. Peraire, and L. Casals. A variationally consistent fractional time-step integration method for incompressibility and nearly incompressible Lagrangian dynamics. Int. J. Num. Meth. Eng., 59:1371–95, 2005.

[93] M.C. Lai, Y.H. Tseng, and H. Huang. An immersed boundary method for interfacial flows with insoluble surfactant. J. of Comput. Phys., 227(15):7279–7293, 2008.

[94] H. Lamb. Hydrodynamics. Cambridge Univ Pr, 1997.

[95] R. Lasseter. Principles of traditional animation applied to 3D computer animation. Comput. Graph. (SIGGRAPH Proc.), pages 35–44, 1987.

[96] D.V. Le, J. White, J. Peraire, K.M. Lim, and B.C. Khoo. An implicit immersed boundary method for three-dimensional fluid-membrane interactions. J. of Comput. Phys., 228(22):8427–8445, 2009.

[97] J. Lenoir and S. Fonteneau. Mixing deformable and rigid-body mechanics simulation. In Comput. Graph. Int., pages 327–334, june 16-19 2004.

[98] A.J. Lew and G.C. Buscaglia. A discontinuous-Galerkin-based immersed boundary method. International Journal for Numerical Methods in Engineering, 76(4):427–454, 2008.

[99] P. Liovic, M. Francois, M. Rudman, and R. Manasseh. Efficient simulation of surface tension-dominated flows through enhanced interface geometry interrogation. J. of Comput. Phys., 2010.

[100] W. Lorensen and H. Cline. Marching cubes: A high-resolution 3D surface construction algorithm. Comput. Graph. (SIGGRAPH Proc.), 21:163–169, 1987.

[101] N. Molino, Z. Bao, and R. Fedkiw. A virtual node algorithm for changing mesh topology during simulation. ACM Trans. Graph. (SIGGRAPH Proc.), 23:385–392, 2004.

[102] N. Molino, R. Bridson, J. Teran, and R. Fedkiw. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In 12th Int. Meshing Roundtable, pages 103–114, 2003.

[103] M. Moore and J. Wilhelms. Collision detection and response for computer animation. Comput. Graph. (Proc. SIGGRAPH 88), 22(4):289–298, 1988.

[104] M. Müller, L. McMillan, J. Dorsey, and R. Jagnow. Real-time simulation of deformation and fracture of stiff materials. In Comput. Anim. and Sim. '01, Proc. Eurographics Wrkshp., pages 99–111. Eurographics Assoc., 2001.

[105] M. Müller, S. Schirm, M. Teschner, B. Heidelberger, and M. Gross. Interaction of fluids with deformable solids. J. Comput. Anim. and Virt. Worlds, 15(3–4):159–171, July 2004.

[106] L. P. Nedel and D. Thalmann. Real time muscle deformations using mass-spring systems. In Proc. Comput. Graph. Int., pages 156–165, 1998.

[107] Y-T. Ng, C. Min, and F. Gibou. An efficient fluid–solid coupling algorithm for single–phase flows. J. of Comp. Phys., 228:8807–8829, 2009.

[108] Y.T. Ng, C. Min, and F. Gibou. An efficient fluid-solid coupling algorithm for single-phase flows. J. of Comput. Phys., 228(23):8807–8829, 2009.

[109] D. Nixon and R. Lobb. A fluid-based soft-object model. IEEE Comput. Graph. Appl., 22(4):68–75, 2002.

[110] R.H. Nochetto and S.W. Walker. A hybrid variational front tracking-level set mesh generator for problems exhibiting large deformations and topological changes. J. of Comput. Phys., 229(18):6243–6269, 2010.

[111] A. Norton, G. Turk, B. Bacon, J. Gerth, and P. Sweeney. Animation of fracture by physical modeling. Vis. Comput., 7(4):210–219, 1991.

[112] E. Oñate, J. Rojek, R. Taylor, and O. Zienkiewicz. Finite calculus formulation for incompressible solids using linear triangles and tetrahedra. Int. J. Num. Meth. Eng., 59:1473–1500, 2004.

[113] J. O'Brien, A. Bargteil, and J. Hodgins. Graphical modeling of ductile fracture. ACM Trans. Graph. (SIGGRAPH Proc.), 21:291–294, 2002.

[114] J. O'Brien and J. Hodgins. Graphical modeling and animation of brittle fracture. In Proc. of SIGGRAPH 1999, pages 137–146, 1999.

[115] J. F. O'Brien, V. B. Zordan, and J. K. Hodgins. Combining active and passive simulations for secondary motion. IEEE Comput. Graph. Appl., 20, 2000.

[116] S. Osher and C.-W. Shu. High order essentially non-oscillatory schemes for Hamilton-Jacobi equations. SIAM J. Num. Anal., 28:902–921, 1991.

[117] C. Peskin. Numerical analysis of blood flow in the heart. J. Comput. Phys., 25:220–252, 1977.

[118] G. Picinbono, H. Delingette, and N. Ayache. Nonlinear and anisotropic elastic soft tissue models for medical simulation. In IEEE Int. Conf. Robot. and Automation, 2001.

[119] F. Andrade Pires, E. de Souza Neto, and D. Owen. On the finite element prediction of damage growth and fracture initiation in finitely deforming ductile materials. Comp. Meth. Appl. Mech. Engng., 193:5223–5256, 2004.

[120] J. Platt and A. Barr. Constraint methods for flexible models. Comput. Graph. (SIGGRAPH Proc.), pages 279–288, 1988.

[121] E. Promayon, P. Baconnier, and C. Puech. Physically-based deformations constrained in displacements and volume. In Eurographics, volume 15. Eurographics Assoc., 1996.

[122] S. Punak and J. Peters. Localized volume preservation for simulation and animation. In Poster, SIGGRAPH Proc. ACM, 2006.

[123] A. Robinson-Mosher, R.E. English, and R. Fedkiw. Accurate tangential velocities for solid fluid coupling. In Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim., pages 227–236, 2009.

[124] A. Robinson-Mosher, C. Schroeder, and R. Fedkiw. A symmetric positive definite formulation for monolithic fluid structure interaction. J. Comput. Phys., 230:1547–66, 2011.

[125] A. Robinson-Mosher, T. Shinar, J.T. Grétarsson, J. Su, and R. Fedkiw. Two-way coupling of fluids to rigid and deformable solids and shells. ACM Trans. on Graphics, 27(3):46:1–46:9, August 2008.

[126] J. Rojek, E. O nate, and R. Taylor. CBS-based stabilization in explicit solid dynamics. Int. J. Num. Meth. Eng., 66:1547–68, 2006.

[127] S. H. Roth, M. Gross, M. H. Turello, and S. Carls. A Bernstein-Bézier based approach to soft tissue simulation. Comput. Graph. Forum (Proc. Eurographics), 17(3):285–294, 1998.

[128] C. Schroeder, W. Zheng, and R. Fedkiw. Implicit surface tension formulation with a lagrangian surface mesh on an eulerian simulation grid. J. Comput. Phys., 2011. in review.

[129] A. Selle, R. Fedkiw, B. Kim, Y. Liu, and J. Rossignac. An Unconditionally Stable MacCormack Method. J. of Sci. Comp., 35(2):350–371, 2008.

[130] A. Selle, J. Su, G. Irving, and R. Fedkiw. Robust high-resolution cloth using parallelism, history-based collisions, and accurate friction. IEEE Trans. on Vis. and Comput. Graph., pages 339–350, 2009.

[131] J. Sethian. A fast marching level set method for monotonically advancing fronts. Proc. Natl. Acad. Sci., 93:1591–1595, 1996.

[132] J. Sethian. Fast marching methods. SIAM Review, 41:199–235, 1999.

[133] X. Shi and S.P. Lim. A LBM–DLM/FD method for 3D fluid–structure interactions. J. Comput. Phys, 226(2):2028–2043, 2007.

[134] T. Shinar, C. Schroeder, and R. Fedkiw. Two-way coupling of rigid and deformable bodies. In SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 95–103, 2008.

[135] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock capturing schemes. J. Comput. Phys., 77:439–471, 1988.

[136] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock capturing schemes II (two). J. Comput. Phys., 83:32–78, 1989.

[137] R.K. Shukla, C. Pantano, and J.B. Freund. An interface capturing method for the simulation of multi-phase compressible flows. J. of Comput. Phys., 2010.

[138] E. Sifakis, T. Shinar, G. Irving, and R. Fedkiw. Hybrid simulation of deformable solids. In Proc. of ACM SIGGRAPH/Eurographics Symp. on Comput. Anim., pages 81–90, 2007.

[139] J. Simo and R. Taylor. Quasi-incompressible finite elasticity in principal stretches: continuum basis and numerical examples. Comput. Meth. in Appl. Mech. and Eng., 51:273–310, 1991.

[140] J.C. Simo and K.K. Wong. Unconditionally stable algorithms for rigid body dynamics that exactly preserve energy and momentum. International Journal for Numerical Methods in Engineering, 31(1):19–52, 1991.

[141] P. Smereka. Semi-implicit level set methods for curvature and surface diffusion motion. J. Sci. Comput., 19(1):439–456, 2003.

[142] P. Smereka. The numerical approximation of a delta function with application to level set methods. J. Comput. Phys., 211:77–90, 2006.

[143] D. E. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with Coulomb friction. In IEEE Int. Conf. on Robotics and Automation, pages 162–169, 2000.

[144] J. Su, C. Schroeder, and R. Fedkiw. Energy stability and fracture for frame rate rigid body simulations. In Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim., pages 155–164, 2009.

[145] M. Sussman and M. Ohta. A stable and efficient method for treating surface tension in incompressible two-phase flow. J. Sci. Comput, 31(4):2447–2471, 2009.

[146] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. J. Comput. Phys., 114:146–159, 1994.

[147] Z. Tan, D. Wang, and Y. Wang. A Jacobian-free-based IIM for incompressible flows involving moving interfaces with Dirichlet boundary conditions. Intl. J. for Num. Meth. in Engng., 83:508–536, 2010.

[148] K.E. Teigen, P. Song, J. Lowengrub, and A. Voigt. A diffuse-interface method for two-phase flows with soluble surfactants. J. of Comput. Phys., 230:375–393, 2010.

[149] J. Teran, S. Blemker, V. Ng, and R. Fedkiw. Finite volume methods for the simulation of skeletal muscle. In Proc. of the 2003 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim., pages 68–74, 2003.

[150] J. Teran, E. Sifakis, G. Irving, and R. Fedkiw. Robust quasistatic finite elements and flesh simulation. Proc. of the 2005 ACM SIGGRAPH/Eurographics Symp. on Comput. Anim., pages 181–190, 2005.

[151] J. Teran, E. Sifakis, S. Salinas-Blemker, V. Ng-Thow-Hing, C. Lau, and R. Fedkiw. Creating and simulating skeletal muscle from the visible human data set. IEEE Trans. on Vis. and Comput. Graph., 11(3):317–328, 2005.

[152] M. Teschner, B. Heidelberger, M. Müller, and M. Gross. A versatile and robust model for geometrically complex deformable solids. In Proc. Comput. Graph. Int., pages 312–319, 2004.

[153] Bernhard Thomaszewski, Andreas Gumann, Simon Pabst, and Wolfgang Strasser. Magnets in motion. ACM Trans. Graphics (Proc. SIGGRAPH Asia), 27(5):162:1–162:9, 2008.

[154] P. Thoutireddy, J. Molinari, E. Repetto, and M. Ortiz. Tetrahedral composite finite elements. Int. J. Num. Meth. Eng., 53:1337–1351, 2002.

[155] A.-K. Tornberg and B. Engquist. Numerical approximations of singular source terms in differential equations. J. Comput. Phys., 200(2):462–488, 2004.

[156] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y. J. Jan. A front-tracking method for the computations of multiphase flow. J. Comput. Phys., 169:708–759, 2001.

[157] J. Tsitsiklis. Efficient algorithms for globally optimal trajectories. In Proc. 33rd Conf. on Decision and Control, pages 1368–1373, 1994.

[158] J. Tsitsiklis. Efficient algorithms for globally optimal trajectories. IEEE Trans. on Automatic Control, 40:1528–1538, 1995.

[159] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible, multifluid flows. J. Comput. Phys., 100:25–37, 1992.

[160] B.P. Van Poppel, O. Desjardins, and J.W. Daily. A ghost fluid, level set methodology for simulating multiphase electrohydrodynamic flows with application to liquid fuel injection. J. of Comput. Phys., 229:7977–7996, 2010.

[161] R. van Zon and J. Schofield. Numerical implementation of the exact dynamics of free rigid bodies. J. Comput. Phys., 225:145–164, 2007.

[162] G. Vilmart. Reducing round-off errors in rigid body dynamics. J. Comput. Phys., 227:7083–7088, 2008.

[163] W. von Funck, H. Theisel, and H.-P. Seidel. Vector field based shape deformations. ACM Trans. Graph. (SIGGRAPH Proc.), 25(3):1118–1125, 2006.

[164] H. Wang and T. Belytschko. Fluid-structure interaction by the discontinuous-galerkin method for large deformations. International Journal for Numerical Methods in Engineering, 77(1), 2009.

[165] H. Wang, J. Chessa, W.K. Liu, and T. Belytschko. The immersed/fictitious element method for fluid–structure interaction: Volumetric consistency, compressibility and thin members. Int. J. Numer. Meth. Engng, 74:32–55, 2008.

[166] X. Wang and W.K. Liu. Extended immersed boundary method using FEM and RKPM* 1. Comp. Meth. Appl. Mech. Engng., 193(12-14):1305–1321, 2004.

[167] R. Weinstein. Simulation and Control of Articulated Rigid Bodies. PhD thesis, Stanford University, June 2007.

[168] R. Weinstein, E. Guendelman, and R. Fedkiw. Impulse-based control of joints and muscles. IEEE Trans. on Vis. and Comput. Graph., 14(1):37–46, 2008.

[169] R. Weinstein, J. Teran, and R. Fedkiw. Dynamic simulation of articulated rigid bodies with contact and collision. IEEE Trans. on Vis. and Comput. Graph., 12(3):365–374, 2006.

[170] J. Weiss, B. Maker, and S. Govindjee. Finite-element implementation of incompressible, transversely isotropic hyperelasticity. Comput. Meth. in Appl. Mech. and Eng., 135:107–128, 1996.

[171] A. Witkin and M. Kass. Spacetime constraints. In Comput. Graph. (Proc. SIGGRAPH '88), volume 22, pages 159–168, 1988.

[172] J.J. Xu and H.K. Zhao. An Eulerian formulation for solving partial differential equations along a moving interface. J. of Sci. Comput., 19(1):573–594, 2003.

[173] G. Yngve, J. O'Brien, and J. Hodgins. Animating explosions. In Proc. SIGGRAPH 2000, volume 19, pages 29–36, 2000.

[174] S.H. Yoon and M.S. Kim. Sweep-based freeform deformations. In Eurographics, volume 25, pages 487–96. Eurographics Assoc., 2006.

[175] Z. Yu. A DLM/FD method for fluid/flexible-body interactions. J. Comput. Phys., 207(1):1–27, 2005.

[176] H. Zhao, J.B. Freund, and R.D. Moser. A fixed-mesh method for incompressible flow–structure systems with finite solid deformations. J. Comput. Phys., 2007.

[177] W. Zheng, J.-H. Yong, and J.-C. Paul. Simulation of bubbles. In SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, pages 325–333, 2006.