Title:
Numerical Simulation of Elastic, Viscoelastic, and Granular Materials

Author:
Gast, Theodore Finn

Acceptance Date:
2016

Series:
UCLA Electronic Theses and Dissertations

Degree:
Ph.D., Mathematics 0540UCLA

Advisor(s):
Teran, Joseph M

Committee:
Terzopoulos, Demetri, Yin, Wotao, Anderson, Christopher R

Permalink:
http://eprints.cdlib.org/uc/item/0vr7h3j7

Abstract:

University of California
Los Angeles

Numerical Simulation of Elastic, Viscoelastic, and Granular Materials

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Mathematics

by

Theodore Finn Gast

2016

Abstract of the Dissertation

Numerical Simulation of Elastic, Viscoelastic, and Granular Materials

by

Theodore Finn Gast

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 2016

Professor Joseph M. Teran, Chair

Practical time steps in today's state-of-the-art simulators typically rely on Newton's method to solve large systems of nonlinear equations. In practice, this works well for small time steps but is unreliable at large time steps at or near the frame rate, particularly for difficult or stiff simulations. Recasting backward Euler as a minimization problem allows Newton's method to be stabilized by standard optimization techniques. The resulting solver is capable of solving even the toughest simulations at the 24Hz frame rate and beyond. Simple collisions can be incorporated directly into the solver through constrained minimization without sacrificing efficiency. Several collision formulations are presented including for self collisions and collisions against scripted bodies, which are designed for the unique demands of this solver. Finally the Material Point Method (MPM) can be formulated to use the solver, and we present formulations for its use for simulating various materials.

For simulating viscoelastic fluids, foams and sponges, we design our discretization from the upper convected derivative terms in the evolution of the left Cauchy-Green elastic strain tensor. We combine this with an Oldroyd-B model for plastic flow in a complex viscoelastic fluid. While the Oldroyd-B model is traditionally used for viscoelastic fluids, we show that its interpretation as a plastic flow naturally allows us to simulate a wide range of complex material behaviors. In order to do this, we provide a modification to the traditional Oldroyd-B model that guarantees volume preserving plastic flows. Our plasticity model is remarkably

simple (foregoing the need for the singular value decomposition (SVD) of stresses or strains). We show that implicit time stepping can be achieved with an optimization based approach and that this allows for high resolution simulations at practical simulation times.

We demonstrate that the Drucker-Prager plastic flow model combined with a Hencky-strain-based hyperelasticity accurately recreates a wide range of visual sand phenomena with moderate computational expense. We use the Material Point Method (MPM) to discretize the governing equations for its natural treatment of contact, topological change and history dependent constitutive relations. The Drucker-Prager model naturally represents the frictional relation between shear and normal stresses through a yield stress criterion. We develop a stress projection algorithm used for enforcing this condition with a non-associative flow rule that works naturally with both implicit (non-optimization based) and explicit time integration. We demonstrate the efficacy of our approach on examples undergoing large deformation, collisions and topological changes necessary for producing modern visual effects.

The dissertation of Theodore Finn Gast is approved.

Demetri Terzopoulos

Wotao Yin

Christopher R. Anderson

Joseph M. Teran, Committee Chair

University of California, Los Angeles

2016

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like profess my sincere thanks to my tireless advisor Professor Joseph Teran, for his continued and dedicated support of my research. I am sincerely grateful to Professor Demetri Terzopoulos for serving on my doctoral committee, and providing me a workplace in his excellent Computer Graphics and Vision Lab. I also thank Professor Christopher Anderson and Professor Wotao Yin for serving on my doctoral committee and helping me through the PhD process.

I thank my coauthors for all their hard work and advice, which was invaluable in the course of my degree. Specifically I would like to thank them for the following contributions. Pirouz Kavehpour for the idea that our method for simulating non-Newtonian fluids could also be applied to foam and sponges. Gergely Klar for the many discussions of sand behavior while implementing Drucker-Prager. Chuyuan Fu for parameter tuning and voicing the sand video. Andre Pradhana for trying so hard to get wet sand to fracture. Daniel Ram for the many late nights we spent hunting for bugs together. Craig Schroeder for his expertise in physics, computer performance and taking derivatives. Chenfanfu Jiang for his exquisite renderings and mastery of MPM. Alexey Stomakhin for his guidance in tuning parameters and advisement during my internship at Disney Animation.

I thank my family for their unconditional love and support throughout my life, my parents, Katie and Ted, my sister Marion, my grandparents and all my aunts, uncles and cousins.

This dissertation contains content from previously published work. Chapter Two from [GSS15], Chapter Three from [RGJ15], Chapter Four from [KGP16b] and [KGP16a].

The author was partially supported by NSF (CCF-1422795), ONR (N000140310071, N000141010730, N000141110719, N000141210834), DOE (09-LR-04-116741-BERA), DOD (W81XWH-15-1-0147), Intel STC-Visual Computing Grant (20112360) as well as a gift from Disney Research.

Vita

2010	B.S. (Mathematics), Carnegie Mellon University.
2011–2016	Teaching Assistant, Mathematics Department, UCLA.
2013–2016	Research Assistant, Mathematics Department, UCLA.
2015	Summer Intern, Walt Disney Animation Studios

Publications

G. Klar, T. F. Gast, A. Pradhana, C. Fu, C. Schroeder, C. Jiang, and J.M. Teran. "Drucker-Prager Elastoplasticity for Sand Animation." ACM Trans. Graph. (SIGGRAPH), 2016

T. F. Gast, C. Schroeder, A. Stomakhin, C. Jiang, and J.M. Teran. "Optimization Integrator for Large Time Steps." Visualization and Computer Graphics, IEEE Transactions on, 21(10):1103-1115, Oct 2015.

D. Ram, T. F. Gast, C. Jiang, C. Schroeder, A. Stomakhin, J. Teran, and P. Kavehpour. "A Material Point Method for Viscoelastic Fluids, Foams and Sponges." In ACM SIGGRAPH / Eurographics Symposium on Computer Animation, pp 157-163, New York, NY, USA, 2015. ACM.

T. F. Gast and C. Schroeder. "Optimization Integrator for Large Time Steps." In Eurographics/ACM SIGGRAPH Symposium on Computer Animation, pp 31-40, Copenhagen, Denmark, 2014. Eurographics Association.

CHAPTER 1

Introduction

Numerical simulation of solid and fluid materials draws on a long history of applied mathematics for the approximation of partial differential equations. In particular the field of computer graphics provides an inspiration for solution techniques that have different requirements from more traditional fields. In particular it requires techniques where robustness is often more important than accurate results, as long as the results are visually realistic. This is because software implementation will often be used by artists with little knowledge of the math and physics involved. Thus techniques that require sophisticated parameter tuning to achieve reliable results are unmanagable. To that end we present an optimization based integration technique, which is robust to poor parameter choices. In particular it is robust to taking large time steps, and in fact the choice of a larger time step than would be allowed by other methods, sometimes provides for a more effiecient solution. Subsequently we present material point methods for the simulation of a variety of materials, including snow, foam, sponges, and sand. All of which are able to take advantage of the robustness of the optimization based integrator with the exception of the sand model as its forces do not come from a potential.

1.1 Optimization based integration

The most commonly used time integration schemes in use today for graphics applications are implicit methods. Among these, backward Euler [BW98, HFL01, VM01, MTG11, LBO13] or variants on Newmark methods [Kan99, BFA02, BMF03] are the most common, though even more sophisticated schemes like BDF-2 [HE01, CK05], implicit-explicit

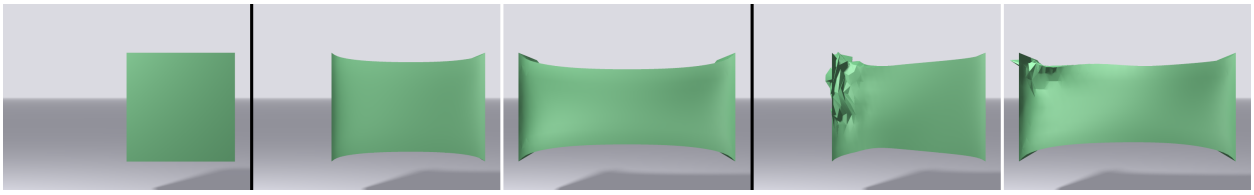


Figure 1.1: *Cube being stretched: initial configuration (left), our method at* $t = 0.4\,s$ *and* $t = 3.0\,s$ *(middle), and standard Newton's method at* $t = 0.4\,s$ *and* $t = 3.0\,s$ *(right). Both simulations were run with one time step per* $24\,Hz$ *frame. Newton's method requires three time steps per frame to converge on this simple example.*

schemes [EEH00, SG09], or even the more exotic exponential integrators [MSW13] have received consideration. Integrators have been the subject of comparison before (see for example [HE01, VM01, PF02]), seeking good compromises between speed, accuracy, robustness, and dynamic behavior.

These integrators require the solution to one or more nonlinear systems of equations each time step. These systems are typically solved by some variation on Newton's method. Even the most stable simulators are typically run several time steps per $24\,Hz$ frame of simulation. There is growing interest in running simulations at larger time steps [SSF13], so that the selection of Δt can be made based on other factors, such as damping or runtime, and not only on whether the simulator works at all. One of the major factors that limits time step sizes is the inability of Newton's method to converge reliably at large time steps (See Figures 1.1, 1.2, and 2.8), or if a fixed number of Newton iterations are taken, the stability of the resulting simulation. We address this by formulating our nonlinear system of equations as a minimization problem, which we demonstrate can be solved more robustly. The idea that dynamics, energy, and minimization are related has been known since antiquity and is commonly leveraged in variational integrators [STW92, KMO99, Kan99, LMO04, KYT06, SG09, GSO10]. The idea that the nonlinear system that occurs from methods like backward Euler can be formulated as a minimization problem has appeared many times in graphics in various forms [HFL01, KYT06, MTG11, LBO13, MSW13]. [KYT06] point out that minimization leads to a method that is both simpler and faster than the equivalent nonlinear root-finding problem, and [LBO13] show that a minimization formulation can be used to solve mass-spring

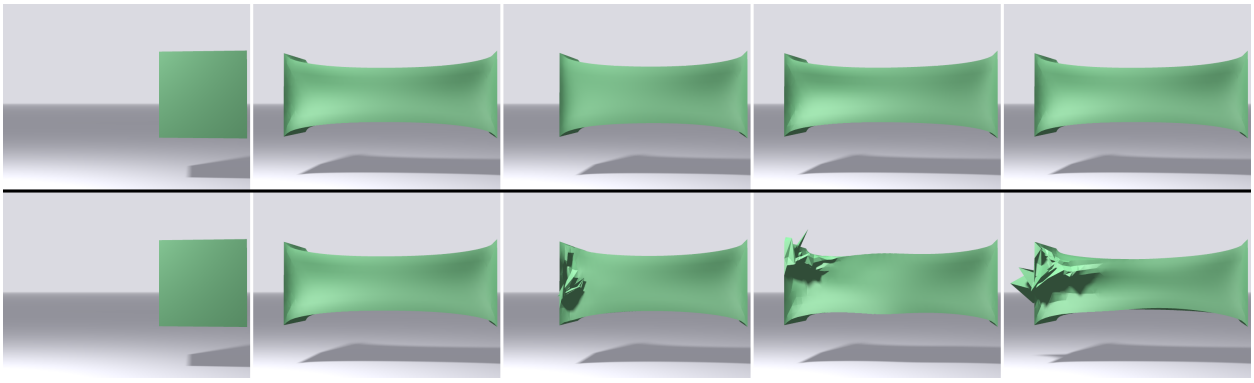


Figure 1.2: *Cube being stretched and then given a small compressive pulse, shown with our method (top) and standard Newton's method (bottom). Both simulations were run with one time step per* $24\,Hz$ *frame. In this simulation, Newton's method is able to converge during the stretch phase, but a simple pulse of compression, as would normally occur due to a collision, causes it to fail to converge and never recover. Newton's method requires five time steps per frame to converge on this simple example.*

systems more efficiently. [KMO99] use a minimization formulation as a means of ensuring that a solution to their nonlinear system can be found assuming one exists. [GHF07] shows that a minimization formulation can be used to enforce constraints robustly and efficiently. [HFL01] shows that supplementing Newton's method with a line search greatly improves robustness. [MTG11] also shows that supplementing Newton's method with a line search and a definiteness correction leads to a robust solution procedure. Following their example, we show that recasting the solution of the nonlinear systems that result from implicit time integration schemes as a nonlinear optimization problem results in substantial robustness improvements. We also show that additional improvements can be realized by incorporating additional techniques like Wolfe condition line searches which curve around collision bodies, conjugate gradient with early termination on indefiniteness, and choosing conjugate gradient tolerances based on the current degree of convergence. In addition we show that the optimization integrator approach to the MPM snow simulator of [SSC13]. This allows us to take much larger time steps than the original method and results in a significant speedup.

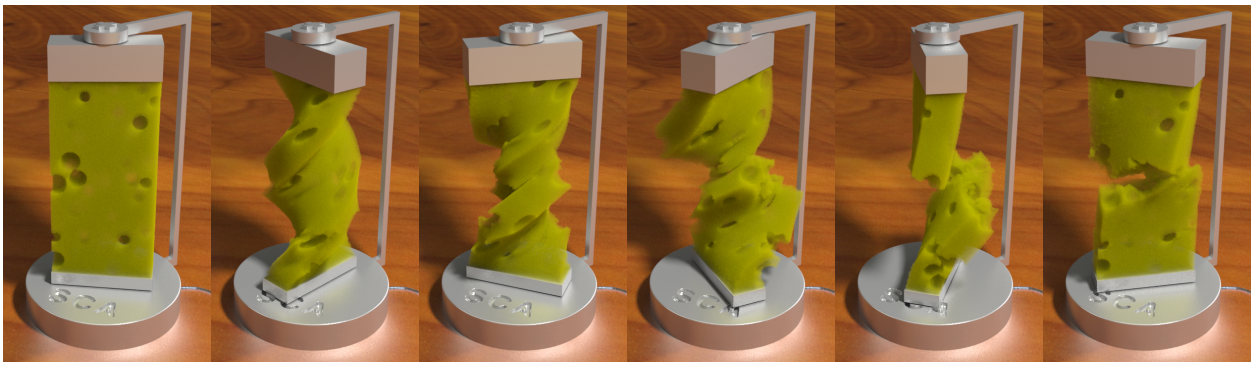


Figure 1.3: *A soft sponge is twisted. It fractures and collides with itself. The failure and contact phenomena are resolved automatically by the MPM approach.*

1.2 Viscoelastic materials

Viscoelastic behavior is exhibited by a wide range of everyday materials including paint, gels, sponges, foams and various food components like ketchup and custard [Lar99]. These materials are often special kinds of colloidal systems (a type of mixture in which one substance is dispersed evenly throughout another), where dimensions exceed those usually associated with colloids (up to 1μm for the dispersed phase) [HR97, Lar99]. For example, when a gas and a liquid are shaken together, the gas phase becomes a collection of bubbles dispersed in the liquid: this is the most common observation of foams. While a standard Newtonian viscous stress is a component of the mechanical response of these materials, they are non-Newtonian in the sense that there are other, often elastoplastic, aspects of the stress response to flow rate and deformation. Comprehensive reviews are given in [MR02, PK96, Sch94, Lar99].

Discretization of these materials is challenging because of the wide range of behaviors exhibited and by the nonlinear governing equations. These materials can behave with elastic resistance to deformation but can also undergo very large strains and complex topological changes characteristic of fluids. While Lagrangian approaches are best for resolving the solid-like behavior and Eulerian approaches most easily resolve the fluid-like behavior, these materials are in the middle ground and this makes discretization difficult. The Material Point Method is naturally suited for this class of materials because it uses a Cartesian grid to resolve topology changes and self-collisions combined with Lagrangian tracking of mass, momentum and deformation on particles. In practice, the particle-wise deformation

information can be used to represent elastoplastic stresses arising from changes in shape, while an Eulerian background grid is used for implicit solves.

We show that the MPM approach in [SSC13] can be generalized to achieve a wide range of viscoelastic, complex fluid effects. As in [SSC13], we show that implicit time stepping can easily be used to improve efficiency and allow for simulation at high spatial resolution. With our Oldroyd-inspired approach, we avoid the need for the SVD of either elastic or plastic responses. While SVD computation is not a bottleneck for MPM when done efficiently (see e.g. [MZS11]), it is not a straightforward implementation. More standard SVD implementations can have a dramatic impact on performance (see e.g. [CPS10]). Thus although it is not essential for performance to avoid the SVD, it is preferable to avoid the need to implement them when, as with our model, they are not necessary for achieving desired behaviors.

1.3 Sand

Sand dynamics are ubiquitous in every day environments. Its characteristic flowing and piling motions must be recreated with a high level of accuracy when animating scenes like beaches or playgrounds. Sand and many other similar everyday materials like salt, powder, rubble, etc are granular materials composed of many discrete macroscopic grains colliding and sliding against one another. These materials exhibit complex behaviors with aspects comparable to both fluids (e.g. they can assume the shape of a container) and solids (they can support weight and form stable piles) [JNB96, BBS00].

Unfortunately, this complex material behavior makes it very difficult to develop numerical methods capable of reproducing sand dynamics. Given the incredibly high number of grains in practical scenarios, a continuum description of the governing equations is useful for simulation. However, it is difficult to design a single constitutive law that reproduces all sand behaviors. Furthermore, these laws are relatively complex and require subtle aspects of elastic and plastic response. On the other hand, a Lagrangian view where all grains are simulated only requires a description of the frictional contact between grains. Unfortunately, numerical methods designed from this view would require the computationally prohibitive

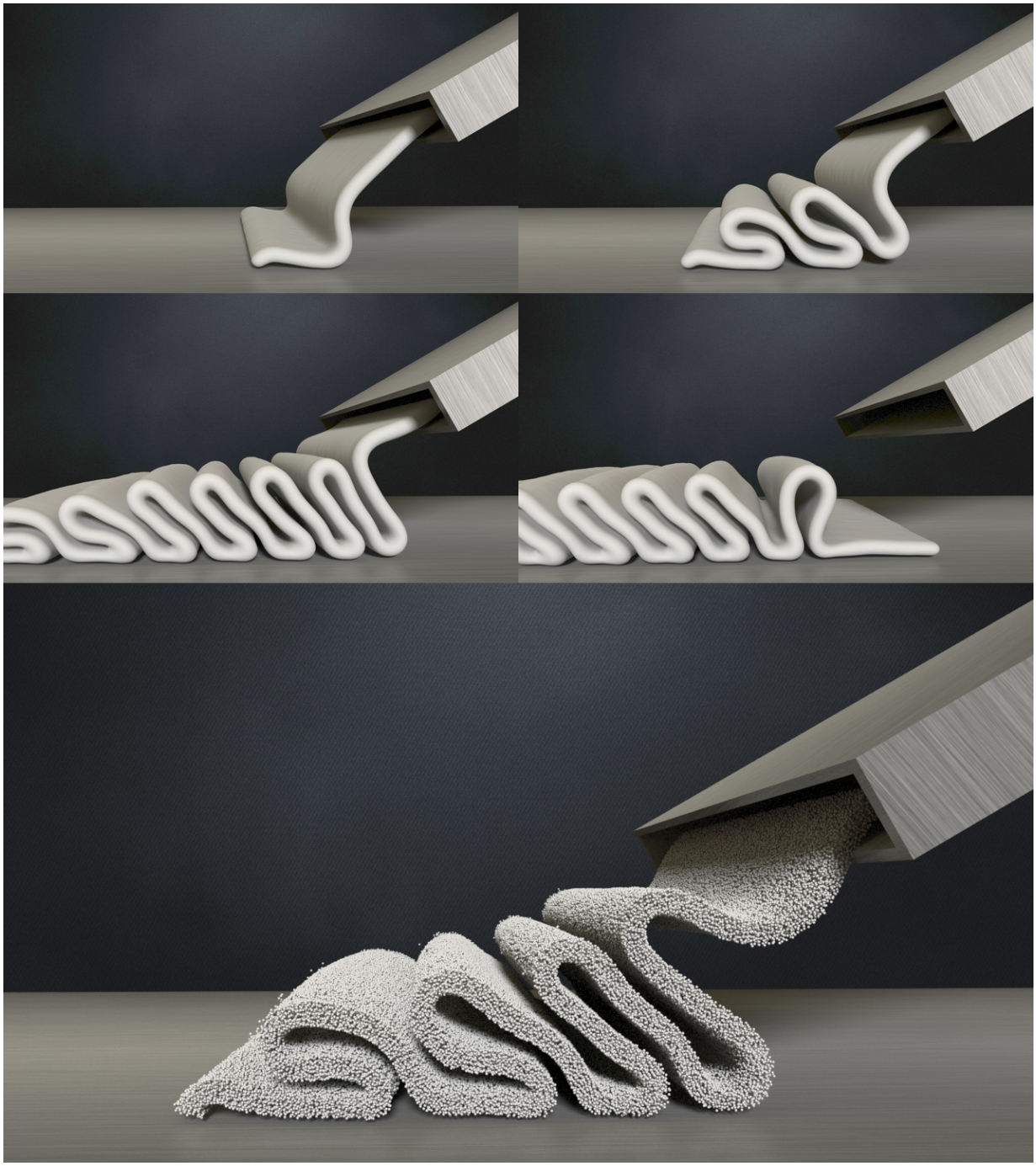


Figure 1.4: *Viennetta ice cream is poured onto a conveyor belt and forms characteristic folds. A particle view is shown on the bottom.*

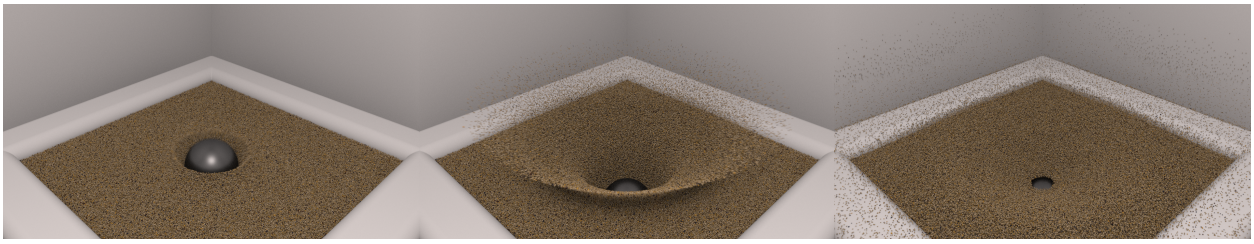


Figure 1.5: *A solid ball drops into a sandbox, spraying sand in all directions.*

simulation of many millions or even of billions of individual grains. Furthermore, it is difficult to tune the grain frictional parameters to match observed piling and flowing behaviors in everyday experiments.

We build on the work of Mast et al. [Mas13, MAM14] and develop an implicit version of their Drucker-Prager-based elastoplasticity model for granular materials. The Drucker-Prager conception of elastoplasticity is often used in the mechanical engineering literature for granular materials [DP52], and we show that it can be adopted to animation applications with relatively simple implementation and efficient runtimes. This is useful because the models are well developed and the literature can be consulted to reduce the difficulty of parameter tuning. We use the Material Point Method (MPM) [SCS94] to discretize the equations since it provides a natural and efficient way of treating contact, topological change and history dependent behavior. Furthermore, we show that this can be done with little more effort than was used for simulating snow dynamics in the MPM approach of Stomakhin et al. [SSC13]. Lastly, we replace the particle/grid transfers used by Mast et al. with APIC transfers [Jia15, JSS15] and show that this allows for more stable behavior, particularly with simulations that have higher numbers of particle per cell.

1.4 Related Work

1.4.1 Viscoelastic materials

Terzopoulos and Fleischer were the first in computer graphics to show the effects possible with simulated elastoplastic materials [TF88a, TF88b]. Since those seminal works, many researchers have developed novel methods capable of replicating a wide range of material

behaviors. Generally, these fall into one of three categories: Eulerian grid, Lagrangian mesh or particle based techniques. In addition to the following discussion, we summarize some aspects of our approach relative to a few representative approaches in Table 3.2.

Goktekin et al. [GBO04] showed that the addition of an Eulerian elastic stress with Von Mises criteria plasticity to the standard level set based simulation of free surface Navier Stokes flows can capture a wide range of viscoelastic behaviors. Losasso et al. also use an Eulerian approach [LIG06]. Rasmussen et al. experiment with a range of viscous effects for level set based free surface melting flows in [REN04]. Batty et al. use Eulerian approaches to efficiently simulate spatially varying viscous coiling and buckling [BB08, BH11]. Carlson et al. also achieve a range of viscous effects in [CMH02].

Lagrangian methods naturally resolve deformation needed for elastoplasticity; however, large strains can lead to mesh tangling for practical flow scenarios and remeshing is required. Bargteil et al. show that this can achieve impressive results in [BWH07]. This was later extended to embedded meshes in [WT08] and further treatment of splitting and merging was achieved in [WTG09]. Batty et al. used a reduced dimension approach to simulate thin viscous sheets with adaptively remeshed triangle meshes in [BUA12].

Ever since Desbrun and Gascuel [DG96] showed that SPH can be used for a range of viscous behavior, particle methods have been popular for achieving complex fluid effects. Like Goktekin et al., Chang et al. [CBL09] also use an Eulerian update of the strain for elastoplasic SPH simulations. Solenthaler et al. show that SPH can be used to compute strain and use this to get a range of elastoplastic effects [SSP07]. Becker et al. show that this can be generalized to large rotational motion in [BIT09]. Gerszewski et al. also update deformation directly on particles [GBB09]. [KAG05] and [MKN04] also add elastic effects into SPH formulations. Paiva et al. use a non-Newtonian model for fluid viscosity in [PPL06] and [PPL09].

Although MPM is a hybrid grid/particle method, particles are arguably the primary material representation. MPM has recently been used to simulate elastoplastic flows to capture snow in [SSC13] and varied, melting materials in [SSJ14]. Yue et al. use MPM to

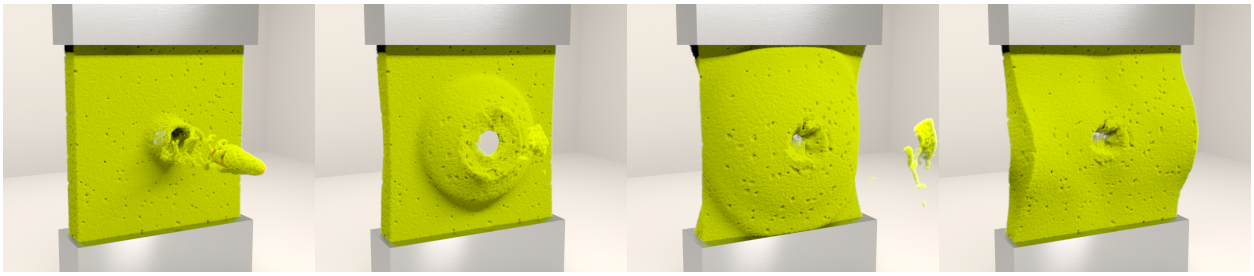


Figure 1.6: *A kinematic bullet is fired at a sponge, resulting in significant deformation and fracture.*

simulate Herschel-Bulkley plastic flows for foam in [YSB15]. Their approach is very similar to ours, however their treatment of plasticity is much more accurate and can handle a wider range of phenomena (notably, shear thickening). They also provide a novel particle splitting technique useful for resolving shearing flows that are problematic for a wide range of MPM simulations. However, their plastic flow update is more complicated and this is likely why they resort to explicit time stepping. With our comparatively simple plastic flow model, we show that semi-implicit time stepping as in [SSC13] can be achieved.

1.4.2 Granular Materials

Continuum approaches have been used in a number of graphics applications. Zhu and Bridson animate sand as a continuum with a modified Particle-In-Cell fluid solver [ZB05]. Narain et al. improve on the method of Zhu and Bridson by removing cohesion artifacts associated with incompressibility [NGL10]. Both of these works led to a number of generalizations and improvements. Nkulikiyimfura et al. [NKK12] develop a GPU version of the Zhu and Bridson approach. Laenerts and Dutre use an SPH version to couple water with porous granular materials [LD09]. Alduan and Otuday [AO11] generalize the unilateral incompressibility developed by Narain et al. to SPH. Imhsen et al. show how to improve the convergence of the method of Alduan and Otaduy [AO11] and also detail refinement of base simulations to upscale to millions of grains [IWT13]. Chang et al. [CBZ12] use a modified Hooke's law to handle friction between grains. MPM is a useful discretization choice for granular and more general elastoplastic flows for computer graphics [YSB15, SSC13, SSJ14, RGJ15, Jia15,

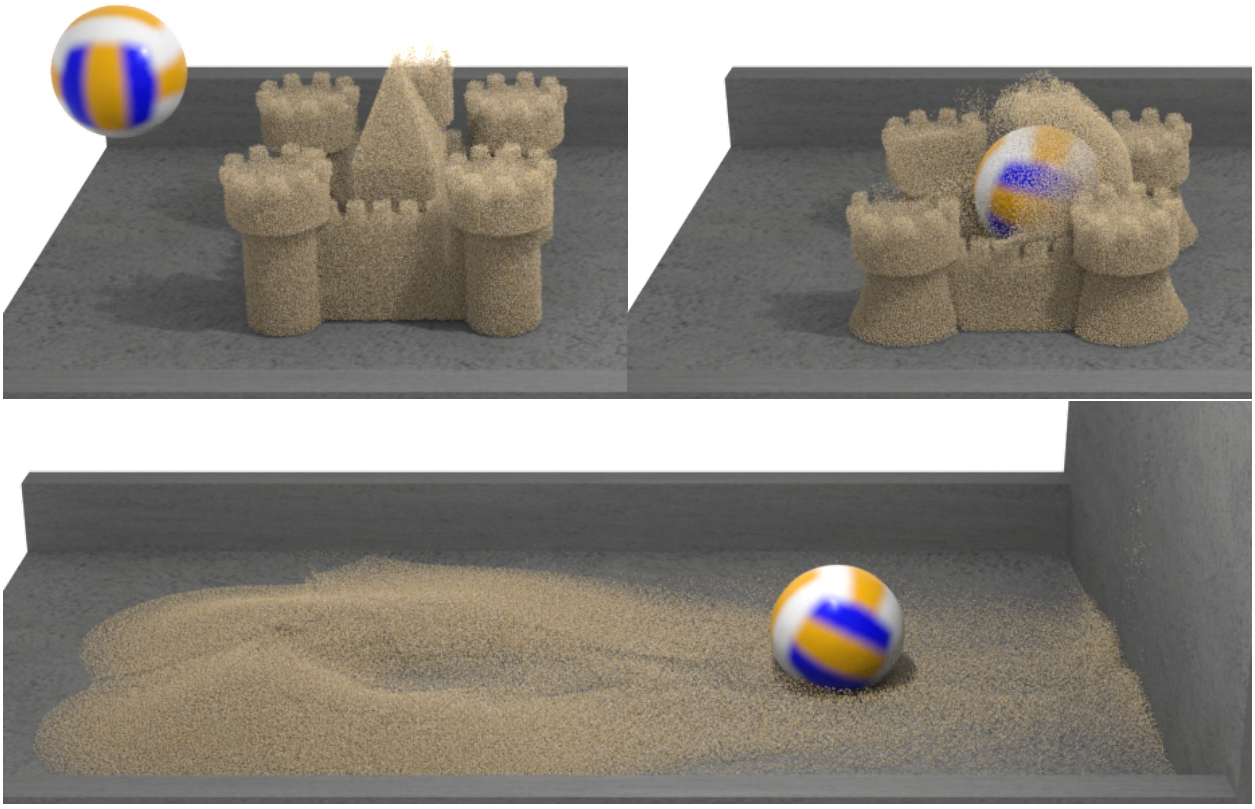


Figure 1.7: *A sand castle is hit with a deformable ball while falling. The sand and ball are fully coupled in the simulation.*

JSS15] and engineering applications [Mas13, MAM14].

Many methods are developed by modeling interactions between individual grains or particle idealizations of grains, rather than from a continuum. Miller and Pearce simulate interactions between particles to model sand, solid and viscous behaviors [MP89]. Luciani et al. use a similar approach [LHM95]. Bell et al. got very impressive results by simulating many sand grains as spherical rigid bodies with friction [BYM05]. Milenkovic also simulated individual grains to solve for piles of rigid materials via energy minimization/optimization [Mil96]. Mazhar et al. use Nestov's method to simulate millions of individual grains [MHN15]. Yasuda et al. use the GPU to get real-time results with rigid grains [YHK08]. Alduan et al. use an adaptive resolution version of the method by Bell et al. [BYM05] to improve performance [ATO09]. Brackbill et al. simulate individual grains but use the MPM to resolve collisions and friction between grains [BBS00, CB02]. Macklin et al. show that the extremely efficient position based dynamics methods can be applied by casting granular interactions as

hard constraints in [MMC14].

When extreme computational efficiency is required, simplified approaches like height fields [SOH99, ON03, LM93, CW13, CLH96] and cellular automata [PGM06] have been used to provide real-time interaction.

1.5 Contributions

We summarize our specific contributions as

- Substantial robustness improvements from recasting the nonlinear systems from implicit time integration schemes as a optimization problem.
- The optimization approach can be applied to the MPM snow simulator which results in a significant speedup.
- A new volume-preserving Oldroyd-B rate-based description of plasticity
- Semi-implicit MPM discretization of viscoelasticity and viscoplasticity, allowing for high spatial resolution simulations
- Rate-based plasticity that does not require an SVD
- An implicit version of MPM Drucker-Prager elastoplasticity model for granular materials.
- Demonstrate APIC transfers [Jia15, JSS15] allow for more stable behavior, particularly with simulations that have higher numbers of particle per cell.

CHAPTER 2

Optimization Integrator

2.1 Time Integration

The equations of motion for simulating solids are

$$\dot{\mathbf{x}} = \mathbf{v} \qquad\qquad \mathbf{M}\dot{\mathbf{v}} = \mathbf{f} \qquad\qquad \mathbf{f} = \mathbf{f}(\mathbf{x}, \mathbf{v}),$$

where $\mathbf{f}$ are forces. As is common in graphics we assume $\mathbf{M}$ is a diagonal lumped-mass matrix. Since we are interested in robustness and large time steps, we follow a backward Euler discretization. This leads to

$$\frac{\mathbf{x}^{n+1} - \mathbf{x}^n}{\Delta t} = \mathbf{v}^{n+1} \qquad\qquad \mathbf{M}\frac{\mathbf{v}^{n+1} - \mathbf{v}^n}{\Delta t} = \mathbf{f}^{n+1} = \mathbf{f}(\mathbf{x}^{n+1}, \mathbf{v}^{n+1}).$$

Eliminating $\mathbf{v}^{n+1}$ yields

$$\mathbf{M}\frac{\mathbf{x}^{n+1} - \mathbf{x}^n - \Delta t\mathbf{v}^n}{\Delta t^2} = \mathbf{f}\left(\mathbf{x}^{n+1}, \frac{\mathbf{x}^{n+1} - \mathbf{x}^n}{\Delta t}\right),$$

which is a nonlinear system of equations in the unknown positions $\mathbf{x}^{n+1}$. This system of nonlinear equations is normally solved with Newton's method. If we define

$$\mathbf{h}(\mathbf{x}^{n+1}) = \mathbf{M}\frac{\mathbf{x}^{n+1} - \mathbf{x}^n - \Delta t\mathbf{v}^n}{\Delta t^2} - \mathbf{f}\left(\mathbf{x}^{n+1}, \frac{\mathbf{x}^{n+1} - \mathbf{x}^n}{\Delta t}\right), \tag{2.1}$$

then our nonlinear problem is one of finding a solution to $\mathbf{h}(\mathbf{x}) = \mathbf{0}$. To do this, one would start with an initial guess $\mathbf{x}^{(0)}$, such as the value predicted by forward Euler. This estimate

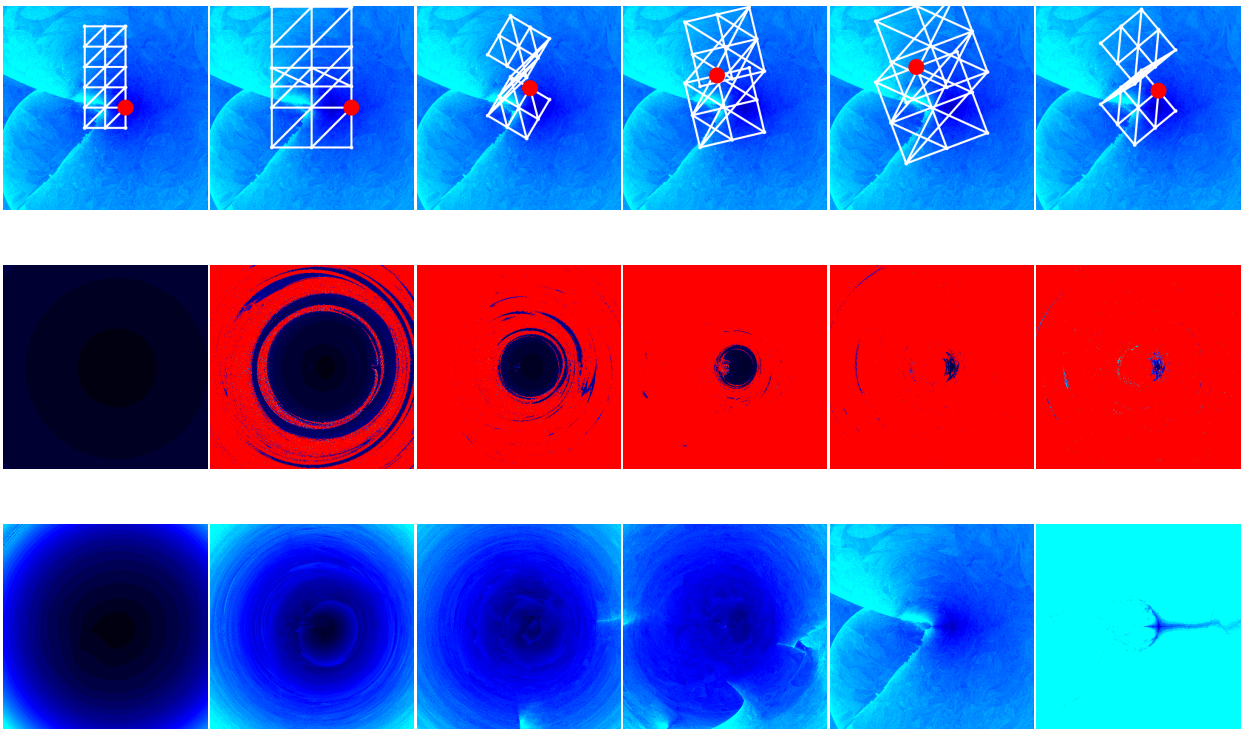


Figure 2.1: *Convergence of Newton's method (middle) and our stabilized optimization formulation (bottom) for a simple 36-dof simulation in 2D. The initial configuration (top) is parameterized in terms of a pixel location, with the rest configuration occurring at* $(\frac{3}{5}, \frac{1}{2})$*. Initial velocity is zero, and one time step is attempted. Time steps are (left to right) 170, 40, 20, 10, and 1 steps per* $24\,Hz$ *frame, with the rightmost image being* $\Delta t = 1\,s$*. Color indicates convergence in 0 iterations (black), 15 iterations (blue), 30 or more iterations (cyan), or failure to converge in 500 iterations (red). Note that Newton's method tends to converge rapidly or not at all, depending strongly on problem difficulty and initial guess.*

is then iteratively improved using the update rule

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}}(\mathbf{x}^{(i)})\right)^{-1} \mathbf{h}(\mathbf{x}^{(i)}).$$

Each step requires the solution of a linear system, which is usually symmetric and positive definite and solved with a Krylov solver such as conjugate gradient or MINRES.

If the function $\mathbf{h}(\mathbf{x})$ is well-behaved and the initial guess sufficiently close to the solution, Newton's method will converge very rapidly (quadratically). If the initial guess is not close enough, Newton's method may converge slowly or not at all. For small enough time steps, the forward and backward Euler time steps will be very similar (they differ by $O(\Delta t^2)$), so a

good initial guess is available. For large time steps, forward Euler will be unstable, so it will not provide a good initial guess. Further, as the time step grows larger, Newton's method may become more sensitive to the initial guess (see Figure 2.1). The result is that Newton's method will often fail to converge if the time step is too large. Figures 1.2, 1.1, and 2.8 show examples of simulations that ought to be routine but where Newton fails to converge at $\Delta t = 1/24\,s$.

Sometimes, only one, or a small fixed number, of Newton steps are taken rather than trying to solve the nonlinear equation to a tolerance. The idea is that a small number of Newton steps is sufficient to get most of the benefit from doing an implicit method while limiting its cost. Indeed, even a single Newton step with backward Euler can allow time steps orders of magnitude higher than explicit methods. Linearizing the problem only goes so far, though, and even these solvers tend to have time step restrictions for tough problems.

2.1.1 Assumptions

We have found that when trying to be very robust, assumptions matter. Before introducing our formulation in detail, we begin by summarizing some idealized assumptions we will make. In practice, we will relax some of these as we go along.

A1: Masses are positive

A2: $\mathbf{f} = -\frac{\partial \Phi}{\partial \mathbf{x}}$ for some function Φ

A3: Φ is bounded from below

A4: Φ is C^1

Assumption (A1) implies that $\mathbf{M}$ is symmetric and positive definite and is useful for theoretical considerations; scripted objects violate this assumption, but they do not cause problems in practice.

Conservative forces always satisfy assumption (A2), and most practical elastic force models will satisfy this. We will show in Section 2.3.2 that even some damping models can be

put into the required form. Friction can be given an approximate potential which is valid for small Δt (See [PKM02]). Since our examples focus on taking larger time steps we address the problem by incorporating friction explicitly after the Newton solve.

Assumption (A3) is generally valid for constitutive models, with the global minimum occurring at the rest configuration. Gravity is an important example of a force that violates this assumption. In Section 2.1.3, we show that assumption (A3) can be safely relaxed to include forces like gravity.

Assumption (A4) is a difficult assumption. Technically, this assumption is a show-stopper, since we know of no constitutive model that is both robust and satisfies it everywhere. To be practical, this must be immediately loosened to C^0, along with a restriction on the types of kinks that are permitted in Φ. The practical aspects of this are discussed in Section 2.2.3.

2.1.2 Minimization problem

The solution to making Newton's method converge reliably is to recast the equation solving problem as an optimization problem, for which robust and efficient methods exist. In principle, that can always be done, since solving $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ is equivalent to minimizing $\|\mathbf{h}(\mathbf{x})\|$ assuming a solution exists. This approach is not very convenient, though, since it requires a global minimum of $\|\mathbf{h}(\mathbf{x})\|$. Further minimization using Newton's method would require the Hessian of $\|\mathbf{h}(\mathbf{x})\|$, which involves the second derivatives of our forces. The standard approach only requires first derivatives. What we really want is a quantity E that we can minimize whose second derivatives only require the first derivatives of our forces. That is, we need to *integrate* our system of nonlinear equations $\mathbf{h}(\mathbf{x})$. Assumption (A2) allows us to do this. This way of recasting the problem also requires only a local minimum be found.

We can write (2.1) as

$$\mathbf{h}(\mathbf{x}) = \mathbf{M}\frac{\mathbf{x} - \mathbf{x}^n - \Delta t\mathbf{v}^n}{\Delta t^2} + \frac{\partial \Phi}{\partial \mathbf{x}}.$$

We note that if we set

$$\hat{\mathbf{x}} = \mathbf{x}^n + \Delta t \mathbf{v}^n \qquad\qquad E(\mathbf{x}) = \frac{1}{2\Delta t^2}(\mathbf{x} - \hat{\mathbf{x}})^T \mathbf{M}(\mathbf{x} - \hat{\mathbf{x}}) + \Phi,$$

then we have $\mathbf{h} = \frac{\partial E}{\partial \mathbf{x}}$. If the required assumptions are met, a global minimum of E always exists.[1] By assumption (A4), $E(\mathbf{x}^{n+1})$ is smooth at its minima, so $\frac{\partial E}{\partial \mathbf{x}}(\mathbf{x}^{n+1}) = \mathbf{0}$ or equivalently $\mathbf{h}(\mathbf{x}^{n+1}) = \mathbf{0}$.[2] Any local minimum is a solution to our original nonlinear equation (2.1). *Although we are now doing minimization rather than root finding, we are still solving exactly the same equations. The discretization and dynamics will be the same, but the solver will be more robust. In particular, we are not making a quasistatic approximation.*

2.1.3 Gravity

A graphics simulation would not be very useful without gravity. Gravity has the potential energy function $-\mathbf{M}\mathbf{g}^T\mathbf{x}$, where $\mathbf{g}$ is the gravitational acceleration vector, but this function is not bounded. An object can fall arbitrarily far and liberate a limitless supply of energy, though in practice this fall will be stopped by the ground or some other object. Adding the gravity force to our nonlinear system yields

$$\mathbf{h}(\mathbf{x}) = \mathbf{M}\frac{\mathbf{x} - \mathbf{x}^n - \Delta t \mathbf{v}^n}{\Delta t^2} - \mathbf{M}\mathbf{g} + \frac{\partial \Phi}{\partial \mathbf{x}},$$

which can be obtained from the bounded minimization objective

$$E(\mathbf{x}) = \frac{1}{2\Delta t^2}(\mathbf{x} - \hat{\mathbf{x}} - \Delta t^2 \mathbf{g})^T \mathbf{M}(\mathbf{x} - \hat{\mathbf{x}} - \Delta t^2 \mathbf{g}) + \Phi.$$

[1] Assumptions (A1) and (A3) ensure that E is bounded from below. Let B be a lower bound on Φ. Then, let $L = \Phi(\hat{\mathbf{x}}) - B + 1$ and Ω be the region where $\frac{1}{2\Delta t^2}(\mathbf{x} - \hat{\mathbf{x}})^T \mathbf{M}(\mathbf{x} - \hat{\mathbf{x}}) \leq L$. Note that Ω is a closed and bounded ellipsoid centered at $\hat{\mathbf{x}}$. E must have a global minimum when restricted to the set Ω since it is a continuous function on a closed and bounded domain. Outside Ω, we have $E(\mathbf{x}) > L + B = E(\hat{\mathbf{x}}) + 1$, so that the global minimum inside Ω is in fact a global minimum over all possible values of $\mathbf{x}$.

[2] Relaxation of assumption (A4) is discussed in Section 2.2.3, where Φ is allowed to have ridge-type kinks. Since these can never occur at a relative minimum, the conclusion here is unaffected.

A more convenient choice of E, and the one we use in practice, is obtained by simply adding the effects of gravity $\Phi_g = -\mathbf{M}\mathbf{g}^T\mathbf{x}$ into Φ. Since all choices E will differ by a constant shift, this more convenient minimization objective will also be bounded from below.

2.2 Minimization

The heart of our simulator is our algorithm for solving optimization problems, which we derived primarily from [NW06], though most of the techniques we apply are well-known. We begin by describing our method as it applies to unconstrained minimization and then show how to modify it to handle the constrained case.

2.2.1 Unconstrained minimization

Our optimization routine begins with an initial guess, $\mathbf{x}^{(0)}$. Each iteration consists of the following steps:

1. * Register active set
2. Compute gradient ∇E and Hessian $\mathbf{H}$ of E at $\mathbf{x}^{(i)}$
3. Terminate successfully if $\|\nabla E\| < \tau$
4. Compute Newton step $\Delta x = -\mathbf{H}^{-1}\nabla E$
5. Make sure Δx is a downhill direction
6. Clamp the magnitude of Δx to ℓ if $\|\Delta x\| > \ell$
7. Choose step size α in direction Δx using a line search
8. Take the step: $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + \alpha\Delta x$
9. * Project $\mathbf{x}^{(i+1)}$

Here, τ is the termination criterion, which controls how accurately the system must by solved. The length clamp ℓ guards against the possibility of the Newton step being enormous

(if $\|\Delta x\| = 10^{100}$, computing $\Phi(\mathbf{x}^{(i)} + \Delta x)$ is unlikely to work well). Its value should be very large. Our line search is capable of choosing $\alpha > 1$, so the algorithm is very insensitive with respect to the choice ℓ. We normally use $\ell = 10^3$. Steps beginning with * are only performed for constrained optimization and will be discussed later. A few of the remaining steps require further elaboration here.

Linear solver considerations: Computing the Newton step requires solving a symmetric linear system. The obvious candidate solver for this is MINRES that can handle indefinite systems, and indeed this will work. However, there are many tradeoffs to be made here. In contrast to a normal Newton solve, an accurate estimate for Δx is not necessary for convergence. Indeed, we would still converge with high probability if we chose Δx to be a random vector. The point of using the Newton direction is that convergence will typically be much more rapid, particularly when the superconvergence of Newton's method kicks in. (Choosing $\Delta x = -\nabla E$ leads to gradient descent, for example, which can display notoriously poor convergence rates.) When the current estimate is far from the solution, the exact Newton direction tends to be little better than a very approximate one. Thus, the idea is to spend little time on computing Δx when $\|\nabla E\|$ is large and more time when it is small. We do this by solving the system to a relative tolerance of $\min(\frac{1}{2}, \sigma\sqrt{\max(\|\nabla E\|, \tau)})$. The $\frac{1}{2}$ ensures that we always reduce the residual by at least a constant factor, which guarantees convergence. The scale σ adjusts for the fact that ∇E is not unitless (we usually use $\sigma = 1$). If our initial guess is naive, we must make sure we take at least one minimization iteration, even if ∇E is very small. Using τ here ensures that we do not waste time solving to a tiny tolerance in this case.

Conjugate gradient: One further optimization is to use conjugate gradient as the solver with a zero initial guess. If indefiniteness is encountered during the conjugate gradient solve, return the last iterate computed. If this occurs on the first step, return the right hand side. If this is done, Δx is guaranteed to be a downhill direction, though it might not be sufficiently downhill for our purposes. In practice, indefiniteness will only occur if far from converged, in which case little time is wasted in computing an accurate Δx that is unlikely to be very useful anyway. Indeed, if the system is detectably indefinite and Δx is computed exactly, it

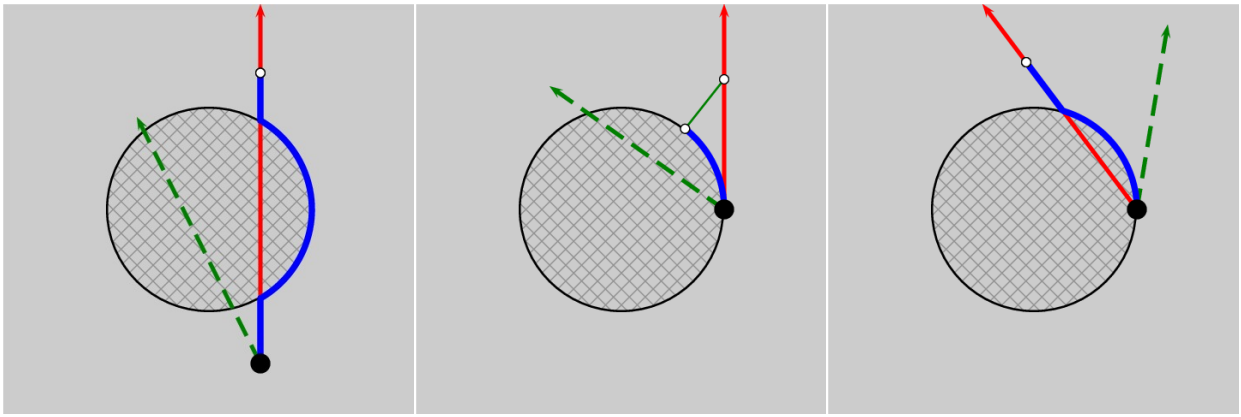


Figure 2.2: *Line search showing the gradient descent direction (green), Newton direction (red), and effective line search path (blue).The constraint is initially feasible (left), active (middle), and touching but inactive (right). Constraints are projected if violated or active, but only inactive constraints may separate.*

might not even point downhill. Since we are searching for a minimum of E (even a local one), the Hessian of E will be symmetric and positive definite near this solution. (Technically, it need only be positive semidefinite, but in practice this is of little consequence.) Thus, when we are close enough to the solution for an accurate Newton step to be useful, conjugate gradient will suffice to compute it. This is very different from the normal situation, where a solver like MINRES or an indefiniteness correction are employed to deal with the possibility of indefiniteness. In the case of our solver, neither strategy is necessary, and both make the algorithm slower.

Downhill direction: Making sure Δx points downhill is fairly straightforward. If $\Delta x \cdot \nabla E < -\kappa \|\Delta x\| \|\nabla E\|$, then we consider Δx to be suitable. Otherwise, if $-\Delta x$ is suitable, use it instead. If neither Δx nor $-\Delta x$ are suitable, then we use the gradient descent direction $-\nabla E$. Note that if the conjugate gradient strategy is used for computing the Newton direction, then $-\Delta x$ will never be chosen as the search direction at this stage. We have found $\kappa = 10^{-2}$ to work well.

Line search: For our line search procedure, we use an algorithm for computing α such that the strong Wolfe Conditions are satisfied. See [NW06] for details. The line search procedure guarantees that E never increases from one iteration to the next and that, provided

certain conditions are met, sufficient progress is always made. One important attribute of this line search algorithm is that it first checks to see if Δx itself is a suitable step. In this way, the line search is almost entirely avoided when Newton is converging properly.

Initial guess: A good initial guess is important for efficient simulation under normal circumstances. Under low-Δt or low-stress conditions, a good initial guess is obtained by replacing $\mathbf{f}^{n+1}$ by $\mathbf{f}^n$ resulting in

$$\mathbf{M}\frac{\mathbf{x}^{(0)} - \mathbf{x}^n - \Delta t\mathbf{v}^n}{\Delta t^2} = \mathbf{f}(\mathbf{x}^n).$$

Solving for $\mathbf{x}^{n+1}$ yields the initial guess

$$\mathbf{x}^{(0)} = \mathbf{x}^n + \Delta t\mathbf{v}^n + \Delta t^2\mathbf{M}^{-1}\mathbf{f}(\mathbf{x}^n).$$

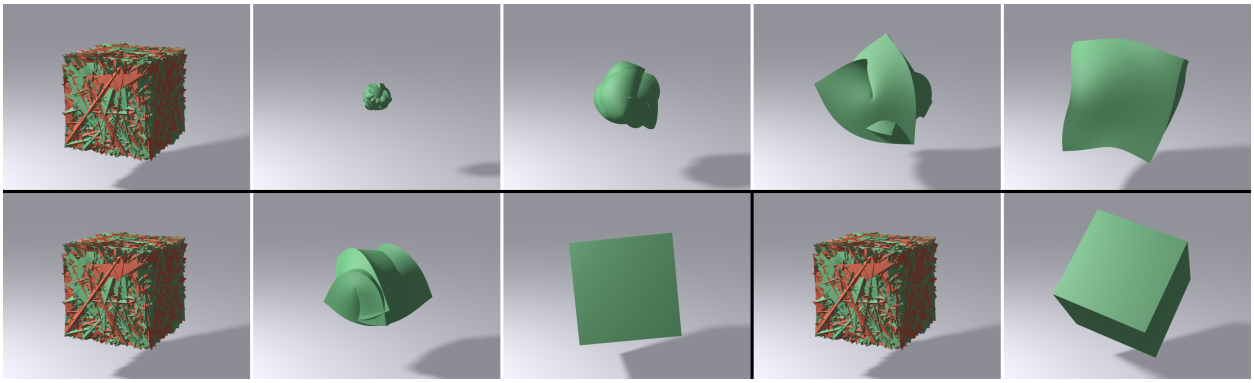


Figure 2.3: *Random test with* $65 \times 65 \times 65$ *particles simulated with* $\Delta t = 1/24\,s$ *for three stiffnesses. Low stiffness recovering over 100 time steps (top), medium stiffness recovering over 40 time steps (bottom left), and high stiffness recovering in a single time step (bottom right). The red tetrahedra are inverted, while the green are uninverted.*

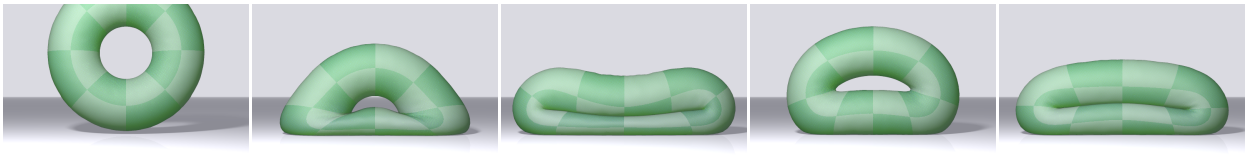


Figure 2.4: *A torus falls on the ground (constraint collisions) and collides with itself (penalty collisions).*

This initial guess is particularly effective under free fall, since here the initial guess is correct and no Newton iterations are required. On the other hand, this initial guess is the result of an explicit method, which will be unstable at large time steps or high stress. Under these conditions, this is unlikely to be a good initial guess and may in fact be very far from the solution. Under these situations, a better initial guess is obtained from $\mathbf{x}^{(0)} = \mathbf{x}^n + \Delta t \mathbf{v}^n$. In practice, we compute both initial guesses and choose the one which produces the smaller value of E. This way, we get competitive performance under easy circumstances and rugged reliability under tough circumstances.

2.2.2 Constrained minimization

We use constrained minimization for some of our collisions, which may result in a large active set of constraints, such as when an ball is bouncing on the ground. As the ball rises, constraints become deactivated. As the ball hits the ground, more constraints become activated. The change in the number of active constraints from iteration to iteration may be quite significant. This would render a traditional active set method impractical, since constraints are activated or deactivated one at a time. Instead, we use the gradient-projection method as our starting point, since it allows the number of active constraints to change

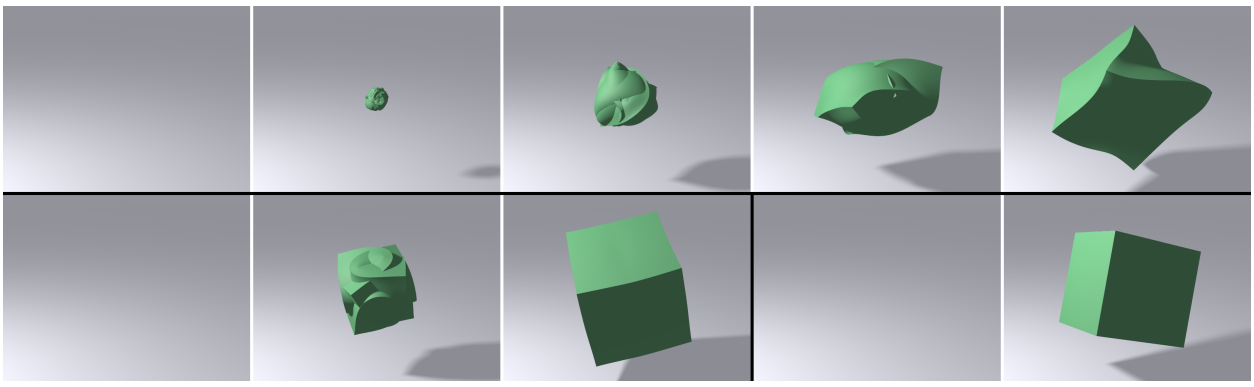


Figure 2.5: *Point test with* $65 \times 65 \times 65$ *particles simulated with* $\Delta t = 1/24\,s$ *for three stiffnesses. Low stiffness recovering over 120 time steps (top), medium stiffness recovering in 5 time steps (bottom left), and high stiffness recovering in a single time step (bottom right).*

quickly. The downside to this choice is that its reliance on the ability to efficiently project to the feasible region limits its applicability to simple collision objects.

Projections: Let $P(\mathbf{x})$ be the projection that applies P_{bp} to $\mathbf{x}_p$ for all body-particle pairs (b, p) that are labeled as active or are violated ($\phi_b(\mathbf{x}_p) < 0$). Note that pairs such that $\phi_b(\mathbf{x}_p) = 0$ (as would be the case once projected) are considered to be touching but not violated. The iterates $\mathbf{x}^{(i)}$ obtained at the end of each Newton step, as well as the initial guess, are projected with P.

Register active set: Let E' be the objective that would be computed in the unconstrained case. The objective function for constrained optimization is $E(\mathbf{x}) = E'(P(\mathbf{x}))$. Compute the gradient $\nabla E'$. Constraints that are touching and for which $\nabla E' \cdot \nabla \phi_b \geq 0$ are labeled as active for the remainder of the Newton step. All others are labeled as inactive. No constraint should be violated at this stage. Note that $E'(\mathbf{x}^{(i)}) = E(\mathbf{x}^{(i)})$ is true before and after every Newton step, since constraints are never violated there.

Curved paths: Note that configurations are always projected to the feasible region before E is computed. One may interpret this as performing line searches along curved paths, as illustrated is Figure 2.2.

When the unprojected line search curve passes through the medial axis of an object, it is possible for the search curve to be disconnected. This causes a discontinuity in the energy as seen from the line search. If the line search does not stop at the discontinuity, the discontinuity has no effect. If it does, the constraint causing the discontinuity will be active (in which case the discontinuity is projected out) or separating (in which case we move away from the discontinuity) in the next Newton step. Thus a disconnected search curve is not a problem for our method.

Discretized level sets: While discontinuities in the curved paths do not pose a problem when the level set is computed correctly, the situation can be quite different when the level set is approximated. This occurs when a grid-based level set is used to approximate a collision object. As a particle moves from cell to cell, the level set approximation (and thus projected location) changes slightly but unpredictably. The resulting kinks or discontinuities

in the search path produce kinks or discontinuities in the objective function along the search line, which may cause the integrator to get stuck. For this reason, we restrict our use of optimization constraints to analytic level sets.

Derivatives: Note also that E must be differentiated twice, and that involves differentiating the projection function P twice. Since P depends on the first derivatives of ϕ_b, the Hessian $\mathbf{H}$ of E would seem to require third derivatives. We note, however, that the only occurrence of the third derivative of ϕ_b occurs multiplied by ϕ_b. Since $\mathbf{H}$ is used only at the beginning of the Newton step when the configuration is feasible, $\phi_b(\mathbf{x}_p) = 0$ or P_{bp} is the identity function. The third derivative term is zero either way, so only the second derivatives of ϕ_b are required.

2.2.3 Practical considerations

There are a few matters of practicality relating to assumption (A4) that are worth mentioning regarding the effective use of this method. The most important of these is that the method does not tolerate discontinuities in E, not even very minute ones, except under some special circumstances that we mention below. In practice, what tends to happen is that a line search encounters a discontinuity in E, where E rises abruptly. The line search dutifully advances the configuration right up to location of this discontinuity. If in the next Newton iteration the descent direction points into the discontinuity, no progress can be made. The solver is stuck.

Discontinuities in ∇E can also cause problems and are impossible to avoid in general. These are kinks in E, which can be broken down into two types: valleys and ridges. The classification is based on whether the kink is ridge-like or valley-like. Ridge-type kinks are acceptable in practice. Valley-type kinks must be avoided, since they can also cause the solver to become stuck for the same reason. A minimum that occurs at a valley-type kink is also problematic since it does not correspond to a solution of (2.1). Thus, the corotated constitutive model, though not completely unusable with this solver, is ill-advised (the fixed variant has no such valleys [SHS12] and is fine). Mass-spring systems are also fine. In

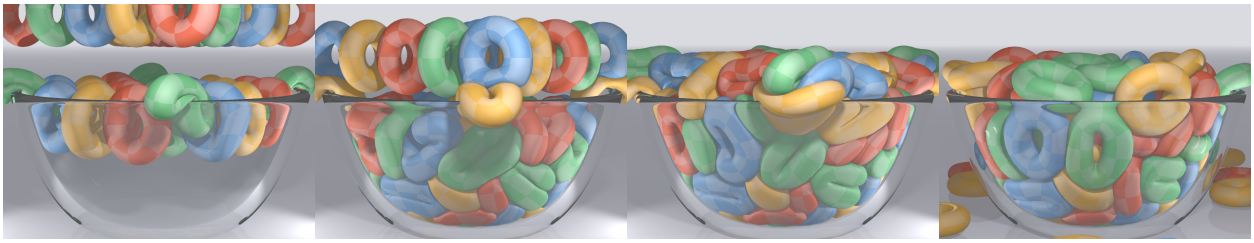


Figure 2.6: *125 tori are dropped into a bowl at 5 time steps per frame, resulting in significant deformation and tough collisions.*

practice, we have only encountered problems when evaluating self-collision models. The self-collision model we propose works well with the method.

The second practical consideration is that E can be somewhat noisy. This is particularly true with forces that involve an SVD, since its computation often involves a balance between speed and accuracy. If the Newton tolerance τ is set too low, the solver will be forced to optimize an objective E where the actual change in E is hidden by the noise. Even with our noisy SVD, we found there is typically at least a three-order-of-magnitude range between the largest value of τ below which no change in output is visually observed and the smallest value above which E is not too noisy to optimize reliably. If we make the E computation robust, E can be optimized down to roundoff level.

Another practical consideration is that occasionally very large changes in the configuration are considered by the line search. For most forces, this is of little consequence. For self-collisions, however, this poses a major performance hazard. We note that when this occurs, the other components of E become very large, too. We first compute all contributions to E except self-collisions. Since our self-collision potential has a global minimum of zero, the real E will be at least as large as the estimate. If this partial E is larger than $E(\mathbf{x}^{(i)})$, we do not compute self-collisions at all. While this presents a discontinuity in E to the optimizer, it is safe to do so under these conditions, since the optimizer will avoid the large value in E by taking a smaller step along the search line.

2.3 Forces

Our formulation is fairly insensitive to the underlying forces, provided it has a continuous potential energy function. We use five forces in our simulations. The simplest of these is gravity, which we addressed in Section 2.1.2. We also employ a hyperelastic constitutive model (Section 2.3.1), a Rayleigh damping model (Section 2.3.2), and two collision penalty force models (Sections 2.4.2 and 2.4.3).

2.3.1 Elastic

A suitable hyperelastic constitutive model must have a few key properties to be suitable for this integrator. The most important is that it must have a potential energy function defined everywhere, and this function must be continuous. The constitutive model must be well-defined for any configuration, including configurations that are degenerate or inverted. This is true even if objects do not invert during the simulation, since the minimization procedure may still encounter such states. Examples of suitable constitutive models are those defined by the corotated hyperelasticity energy [ST08, ZST10, MG04, EKS03, CPS10, MZS11] (but see Section 2.2.3), and the fixed corotated hyperelasticity variant [SHS12]. Stress-based extrapolated models [ITF04, TSI05] are unsuitable due to the lack of a potential energy function in the extrapolated regime, but energy-based extrapolation models [SHS12] are fine. We use the fixed corotated variant [SHS12] for all of our simulations for its combination of simplicity and robustness.

2.3.2 Damping

At first, one might conclude that requiring a potential energy may limit our method's applicability, since damping forces cannot be defined by a potential energy function. A very simple damping model is given by $\mathbf{f} = -k\mathbf{M}\mathbf{v}^{n+1}$. Eliminating the velocity from the equation

yields

$$\mathbf{f}(\mathbf{x}^{n+1}) = -k\mathbf{M}\frac{\mathbf{x}^{n+1} - \mathbf{x}^n}{\Delta t} \qquad (k > 0).$$

The scalar function

$$\Phi(\mathbf{x}^{n+1}) = \frac{k}{2\Delta t}(\mathbf{x}^{n+1} - \mathbf{x}^n)^T\mathbf{M}(\mathbf{x}^{n+1} - \mathbf{x}^n)$$

has the necessary property that $\mathbf{f} = -\frac{\partial\Phi}{\partial\mathbf{x}}$. Note that this Φ looks very similar to our inertial term in E, and it is similarly bounded from below. That this Φ is not a real potential energy function is evident from its dependence on $\mathbf{x}^n$ and Δt, but it is nevertheless suitable for use in our integrator. This simple drag force is not very realistic, though, so we do not use it in our simulations.

A more realistic damping force is Rayleigh damping. Let ψ be an elastic potential energy function. The stiffness matrix corresponding to this force is $-\frac{\partial^2\psi}{\partial\mathbf{x}\partial\mathbf{x}}$, and the Rayleigh damping force and associated objective are

$$\mathbf{f} = -k\left(\frac{\partial^2\psi}{\partial\mathbf{x}\partial\mathbf{x}}(\mathbf{x}^{n+1})\right)\mathbf{v}^{n+1} \qquad \Phi_c = \frac{k}{\Delta t}\left((\mathbf{x}^{n+1} - \mathbf{x}^n)^T\frac{\partial\psi}{\partial\mathbf{x}} - \psi\right).$$

This candidate Φ_c has at least two serious problems. The first is that second derivatives of Φ_c involve third derivatives of ψ. The second is that $\frac{\partial^2\psi}{\partial\mathbf{x}\partial\mathbf{x}}$ may be indefinite, in which case the damping force may not be entirely dissipative. Instead, we approximate Rayleigh damping with a lagged version. Let $\mathbf{D} = \frac{\partial^2\psi}{\partial\mathbf{x}\partial\mathbf{x}}(\mathbf{x}^n)$. Since $\mathbf{D}$ does not depend on $\mathbf{x}^{n+1}$, the lagged Rayleigh damping force and associated objective are

$$\mathbf{f} = -k\mathbf{D}\mathbf{v}^{n+1} \qquad \Phi_d = \frac{k}{2\Delta t}(\mathbf{x}^{n+1} - \mathbf{x}^n)^T\mathbf{D}(\mathbf{x}^{n+1} - \mathbf{x}^n).$$

This solves the first problem, since the second derivative of Φ_d is just $\frac{k}{\Delta t}\mathbf{D}$. Since $\mathbf{D}$ is not being differentiated, it is safe to modify it to eliminate indefiniteness as described in [TSI05, SHS12]. This addresses the second problem. We did not use the damping model

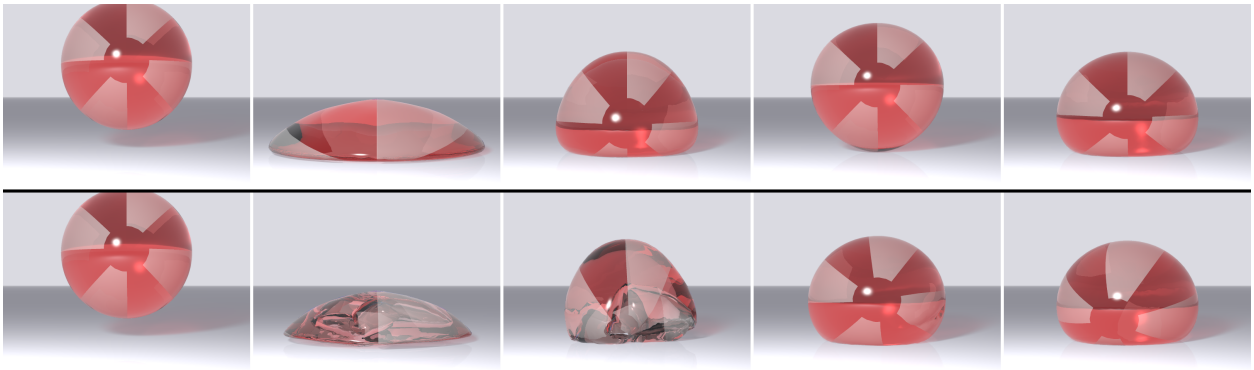


Figure 2.7: *Sphere dropping hard on the ground with $\Delta t = 1/24\,s$ with constraint collisions (top) and collisions as a post-process (bottom). Penalty collisions produce a result very similar to constraint collisions, though some penetration with the ground occurs. Note that the post-processing approach leads to inversion during recovery from the collision.*

found in [KYT06], which uses $\psi(\mathbf{x}^{n+1})$ with $\mathbf{x}^n$ used as the rest configuration, because it is not defined when $\mathbf{x}^n$ is degenerate.

2.4 Collisions

Collisions are a necessary part of any practical computer graphics simulator. The simplest approach to handling collisions is to process them as a separate step in the time integration scheme. This works well for small time steps, but it causes problems when used with large time steps as seen in Figure 2.8. Such arrangement often leads to the collision step flattening objects to remove penetration and the elastic solver restoring the flattened geometry by pushing it into the colliding object. To get around this problem, the backward Euler solver needs to be aware of collisions. A well-tested strategy for doing this is to use penalty collisions, and we do this for two of our three collision processing techniques.

2.4.1 Object collisions as constraints

Our first collision processing technique takes advantage of our minimization framework to treat collisions with non-simulated objects as inequality constraints. Treating collisions or contacts as constraints is not new and in fact forms the basis for LCP formulations such

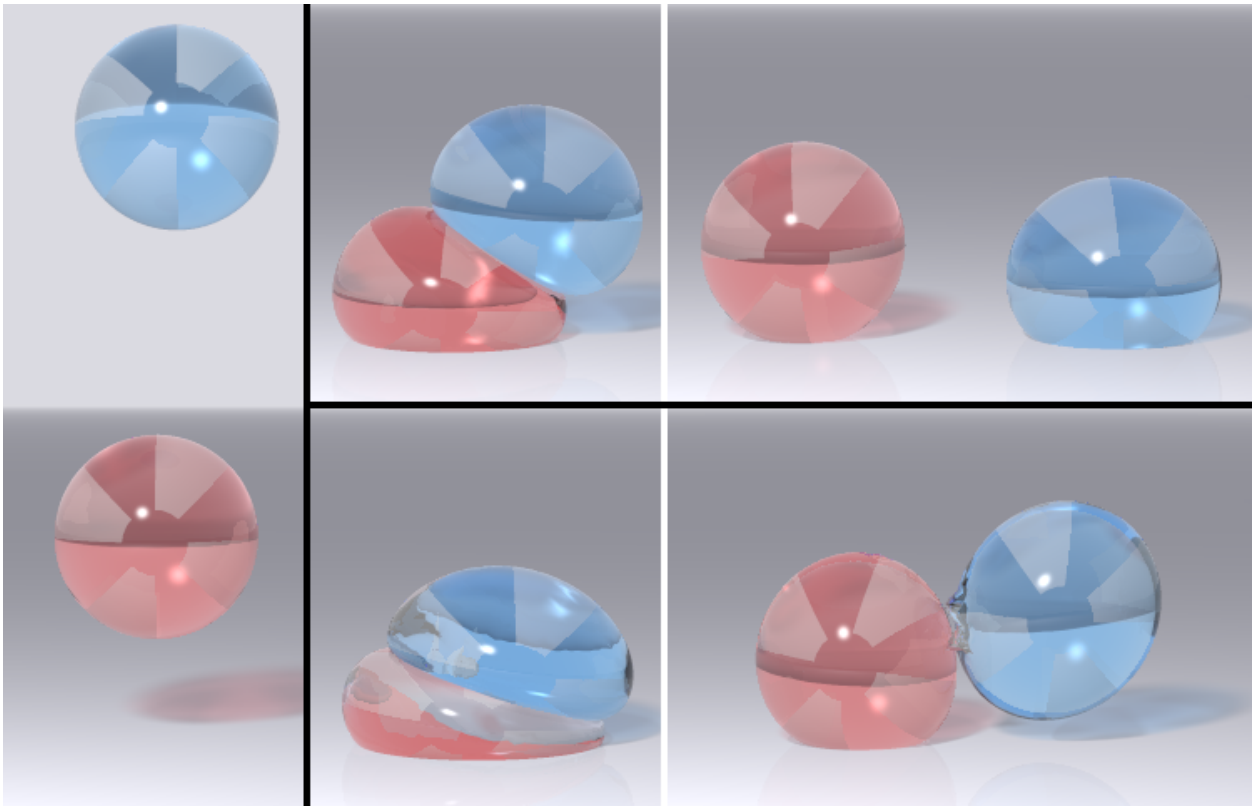


Figure 2.8: *Two spheres fall and collide with one another with $\Delta t = 1/24\,s$: initial configuration (left), our method (top), and Newton's method (bottom). Notice the artifacts caused by Newton not converging. Newton's method requires six time steps per frame to converge on this example.*

as [KSJ08, GZO10]. Unlike LCP formulations, however, our formulation does not attempt to be as complete and as a result can be solved about as efficiently as a simple penalty formulation.

Our constraint collision formulation works reliably when the level set is known analytically. This limits its applicability to analytic collision objects. While this approach is feasible only under limited circumstances, these circumstances occur frequently in practice. When this approach is applicable, it is our method of choice, since it produces better results (e.g., no interpenetration) for similar cost. When this formulation is not applicable, we use a penalty collision formulation instead.

We begin by representing our collision objects (indexed with b) by a level set, which we

denote ϕ_b to avoid confusion with potential energy. By convention, $\phi_b(\mathbf{x}) < 0$ for points $\mathbf{x}$ in the interior of the collision object b. Our collision constraint is simply that $\phi_b(\mathbf{x}_p^{n+1}) \geq 0$ for each simulation particle p and every constraint collision object b. With such a formulation, we can project a particle at $\mathbf{x}_p$ to the closest point $\mathbf{x}_p'$ on the constraint manifold using

$$\mathbf{x}_p' = P_{bp}(\mathbf{x}_p) = \mathbf{x}_p - \phi_b(\mathbf{x}_p)\nabla\phi_b(\mathbf{x}_p).$$

We show how to solve the resulting minimization problem in Section 2.2.2.

We apply friction after the Newton solve. The total collision force felt by particles is

$$\Delta t\mathbf{f}_{col} = \nabla E'(\mathbf{x}^{n+1}) - \nabla E(\mathbf{x}^{n+1}) = \nabla E'(\mathbf{x}^{n+1}) - \nabla E'(P(\mathbf{x}^{n+1})),$$

where E' is the objective in the absence of constraints (See Section 2.2.2). Only collision pairs that are active at the end of the minimization will be applying such forces. We use the level set's normal and the collision force to apply Coulomb friction to colliding particles. In particular, we use the rule $(\mathbf{v}_p^{n+1} \to \hat{\mathbf{v}}_p^{n+1})$

$$\mathbf{n} = \nabla\phi \qquad \mathbf{v}_{p,n}^{n+1} = (\mathbf{n}\cdot\mathbf{v}_p^{n+1})\mathbf{n} \qquad \mathbf{v}_{p,t}^{n+1} = \mathbf{v}_p^{n+1} - \mathbf{v}_{p,n}^{n+1}$$

$$\hat{\mathbf{v}}_p^{n+1} = \mathbf{v}_{p,n}^{n+1} + \max\left(1 - \frac{\mu\Delta t(\mathbf{n}\cdot\mathbf{f}_{p,col})}{m\|\mathbf{v}_{p,t}\|}, 0\right)\mathbf{v}_{p,t}^{n+1}.$$

Our constraint collision formulation is not directly applicable to grid-based level sets, since we assume that $P_{bp}(P_{bp}(\mathbf{x}_p)) = P_{bp}(\mathbf{x}_p)$ and $P_{bp}(x)$ is continuous. Continuity of $P_{bp}(x)$ can be achieved, for example, with C^1 cubic spline level set interpolation. However, it will not generally be true that $P_{bp}(P_{bp}(\mathbf{x}_p)) = P_{bp}(\mathbf{x}_p)$. Alternatively, the projection routine can be modified to iterate the projection to convergence, but then continuity is lost.

2.4.2 Object penalty collisions

When a collision object is not analytic, as will normally be the case for characters for instance, we use a penalty formulation instead. As in the constraint formulation, we assume our collision object is represented by a level set ϕ_b. The elastic potential energy $\Phi_{bp}(\mathbf{x}_p)$ of our penalty force is $\Phi_{bp}(\mathbf{x}) = 0$ if $\phi_b(\mathbf{x}_p) > 0$ and $\Phi_{bp}(\mathbf{x}_p) = k\phi_b(\mathbf{x}_p)^3$ otherwise. Since Φ_{bp} is a potential energy, we must differentiate it twice for our solver. It is important to compute the derivatives of ϕ_b exactly by differentiating the interpolation routine rather than approximating them using central differences. While a C^1 cubic spline interpolation is probably a wiser interpolation strategy since it would avoid the energy kinks that may be caused by a piecewise linear encoding of the level set, we found linear interpolation to work well, too, and we use linear interpolation in our examples.

As in the constraint case, we apply friction after the Newton solve. The total collision force felt by a particle due to object penalty collisions is obtained by evaluating the penalty force at $\mathbf{x}^{n+1}$ and using this force as the normal direction. That is,

$$\mathbf{f}_{col} = -\frac{\partial \Phi_{bp}}{\partial \mathbf{x}}(\mathbf{x}^{n+1}) \qquad f_{p,n} = \|\mathbf{f}_{p,col}\| \qquad \mathbf{n} = \frac{\mathbf{f}_{p,col}}{f_{p,n}}$$

$$\mathbf{v}_{p,n}^{n+1} = (\mathbf{n} \cdot \mathbf{v}_p^{n+1})\mathbf{n} \qquad \mathbf{v}_{p,t}^{n+1} = \mathbf{v}_p^{n+1} - \mathbf{v}_{p,n}^{n+1}$$

$$\hat{\mathbf{v}}_p^{n+1} = \mathbf{v}_{p,n}^{n+1} + \max\left(1 - \frac{\mu \Delta t f_{p,n}}{m\|\mathbf{v}_{p,t}\|}, 0\right)\mathbf{v}_{p,t}^{n+1}.$$

2.4.3 Penalty self-collisions

We detect self-collisions by performing point-tetrahedron inclusion tests, which we accelerate with a bounding box hierarchy. If a point is found to be inside a tetrahedron but not one of the vertices of that tetrahedron, then we flag the particle as colliding.

Once we know a particle is involved in a self collision, we need an estimate for how close the particle is to the boundary. If this particle has collided before, we use the primitive it last collided with as our estimate. Otherwise, we compute the approximate closest primitive in the rest configuration using a level set and use the current distance to this surface element

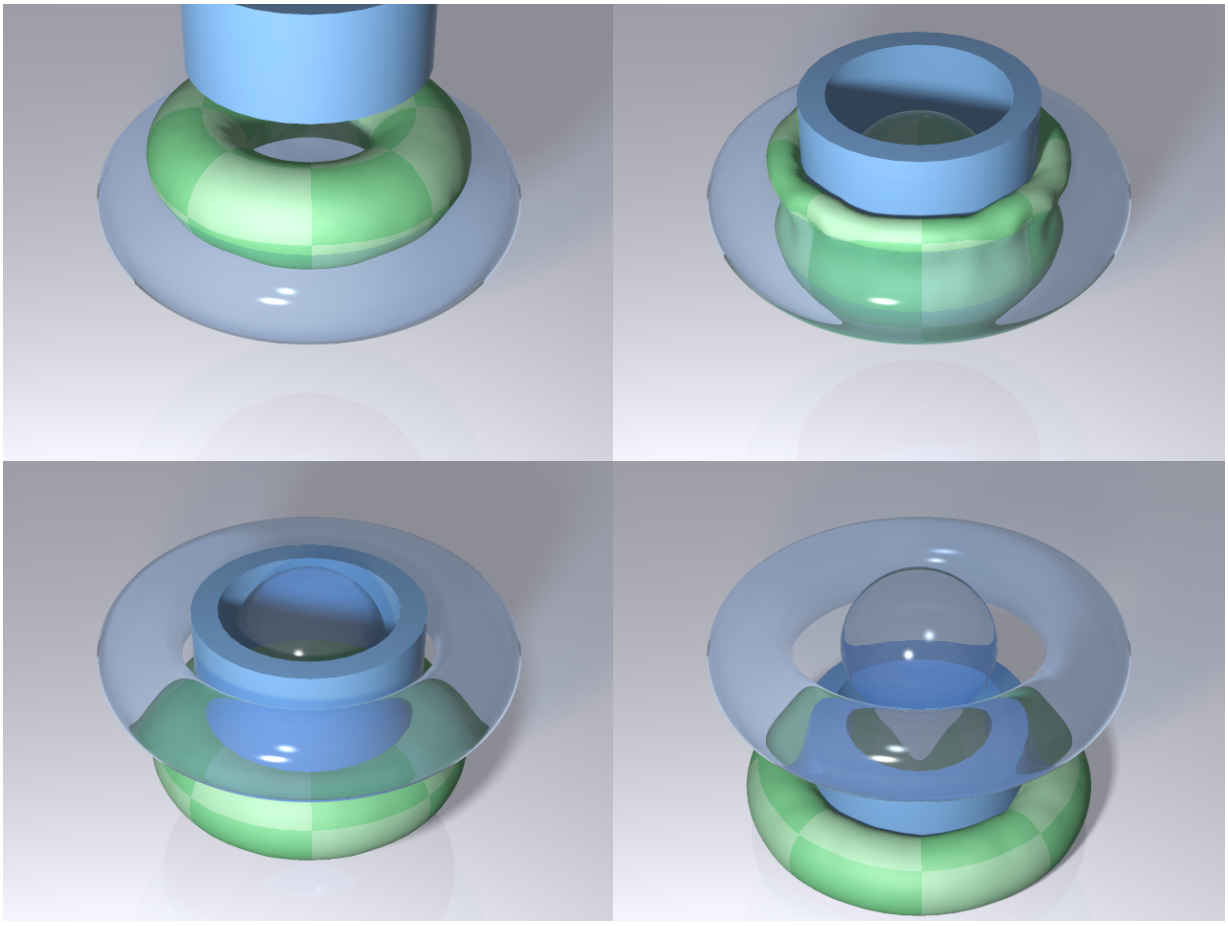


Figure 2.9: *A torus is pushed through a hole (constraint collisions).*

as an estimate.

Given this upper bound estimate of the distance to the boundary, we perform a bounding box search to conservatively return all surface primitives within that distance. We check these candidates to find the closest one. Now we have a point-primitive pair, where the primitive is the surface triangle, edge, or vertex that is closest to the point being processed. Let d be the square of the point-primitive distance. The penalty collision energy for this point is $\Phi = kd\sqrt{d+\epsilon}$, where ϵ is a small number (10^{-15} in our case) to prevent the singularities when differentiating. Note that this penalty function is approximately cubic in the penetration depth. This final step is the only part that must be differentiated.

As with the other two collision models, we apply friction after the Newton solve. In the

most general case, a point n_0 collides with a surface triangle with vertices n_1, n_2, and n_3. As with the object penalty collision model, collision forces are computed by evaluating $\Phi(\mathbf{x}^{n+1})$ and its derivative. The force applied to n_0 is denoted $\mathbf{f}$; its direction is taken to be the normal direction $\mathbf{n}$. The closest point on the triangle to n_0 has barycentric weights w_1, w_2, and w_3. Let $w_0 = -1$ for convenience. Let $\mathbf{Q} = \mathbf{I} - \mathbf{n}\mathbf{n}^T$, noting than $\mathbf{Q}^2 = \mathbf{Q}$. If we apply a tangential impulse $\mathbf{Qj}$ to these particles, their new velocities and kinetic energy will be

$$\hat{\mathbf{v}}_{n_i}^{n+1} = \mathbf{v}_{n_i}^{n+1} + w_i m_{n_i}^{-1} \mathbf{Q}\mathbf{j} \qquad KE = \sum_{n=0}^{3} \frac{1}{2} m_{n_i} (\hat{\mathbf{v}}_{n_i}^{n+1})^T \hat{\mathbf{v}}_{n_i}^{n+1}.$$

We want to minimize this kinetic energy to prevent friction from causing instability. Since M is positive definite, we see that KE is minimized when

$$\nabla KE = \mathbf{Q}\overline{\mathbf{v}} + \overline{m}^{-1}\mathbf{Q}\mathbf{j} = 0 \qquad \overline{\mathbf{v}} = \sum_{n=0}^{3} w_i \mathbf{v}_{n_i}^{n+1} \qquad \overline{m}^{-1} = \sum_{n=0}^{3} w_i m_{n_i}^{-1} w_i.$$

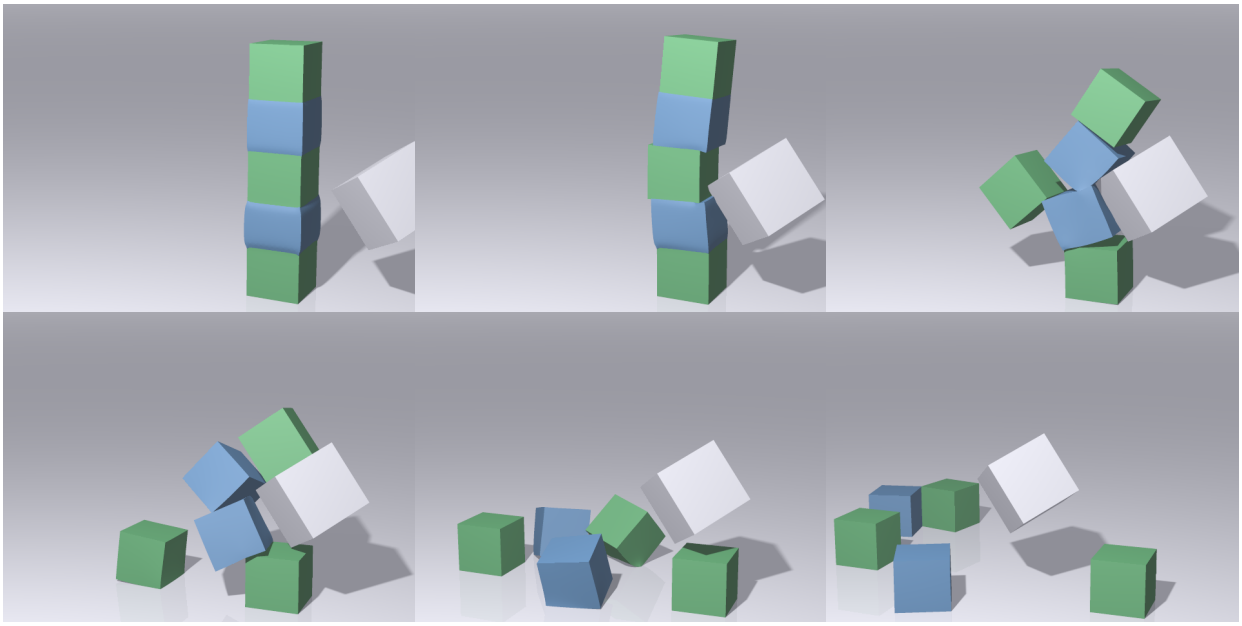


Figure 2.10: *A stack of deformable boxes of varying stiffness is struck with a rigid kinematic cube (constraint collisions) with $\Delta t = 1/24\,s$. The green boxes are 10 times as stiff as the blue boxes.*

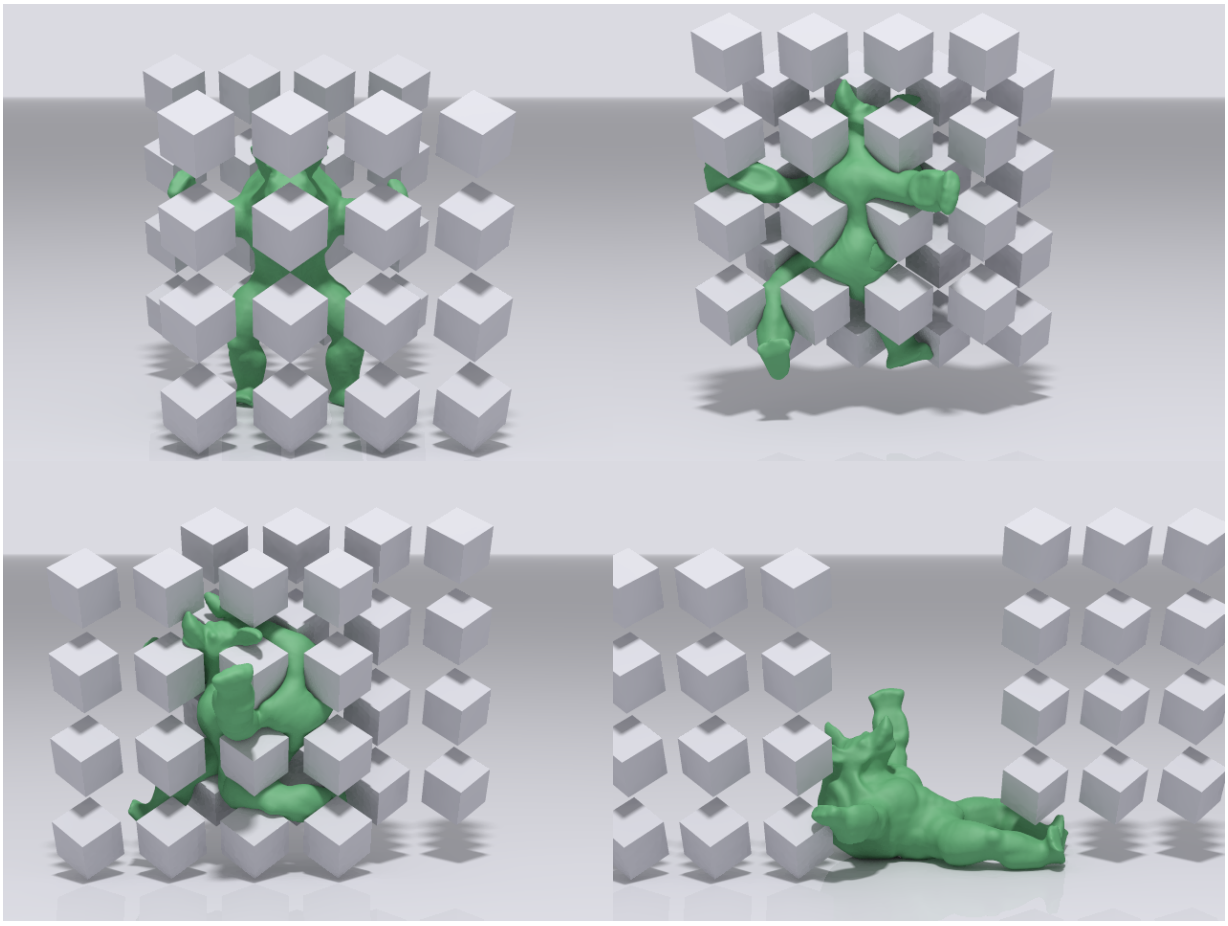


Figure 2.11: *An armadillo is squeezed between 32 rigid cubes (constraint collisions) with* $\Delta t = 1/24\,s$. *When this torture test is run at 1, 2, 4 and 8 steps per frame the average runtime per frame is 46, 58, 88, and 117 seconds respectively.*

If we let $\mathbf{j} = -\overline{m}\mathbf{Q}\overline{\mathbf{v}}$ then $\nabla KE = 0$ and $\mathbf{Q}\mathbf{j} = \mathbf{j}$. This leads to the friction application rule

$$\hat{\mathbf{v}}_{n_i}^{n+1} = \mathbf{v}_{n_i}^{n+1} + w_i m_{n_i}^{-1} \min\left(\frac{\mu\|\mathbf{f}\|}{\|\mathbf{j}\|}, 1\right)\mathbf{j}.$$

Note that all three friction algorithms decrease kinetic energy but do not modify positions, so none of them can add energy to the system, and thus stability ramifications are unlikely even though friction is applied explicitly. This approach to friction can have artifacts, however, since friction will be limited to removing kinetic energy from colliding particles. This limits the amount of friction that can be applied at large time steps. An approach similar to the one in [KSJ08] that uses successive Quadratic Programming solves could possibly be applied

Figure 2.12: *Our approach works naturally with the material point method simulations from [SSC13]. Here we demonstrate with a snowball that drops to the ground and fractures. Notably, we provide a new treatment of particle position updates that naturally prevents penetration in solid objects like the ground.*

to eliminate these artifacts. However [ZJ11] found existing large-scale sparse QP solvers to be insufficiently robust, and thus we did not use this method.

2.5 Accelerating material point method

In this section we describe the application of this optimization approach to the snow simulation from [SSC13]. Their approach to simulating snow uses the material point method (MPM), a hybrid Eulerian-Lagrangian formulation that uses unstructured particles as the primary representation and a background grid for applying forces. They used an energy-based formulation to facilitate a semi-implicit treatment of MPM. While this leads to a significant time step improvement over more standard explicit treatments, it still requires a small time step in practice to remain stable. We show how to modify their original formulation so that we are able to take time steps on the order of the CFL condition. We also provide an improved treatment of collisions with solid bodies that naturally handles them as constraints in the optimization. Although the optimization solve is for grid velocities, we show that a backward Euler (rather than forward Euler) update of particle positions in the grid based velocity field automatically guarantees no particles penetrate solid bodies. In addition to the significantly improved stability, we demonstrate in Section 2.6.1 that in many cases a worthwhile speedup can be obtained with our new formulation.

2.5.1 Revised MPM time integration

In Section 4.1 of [SSC13], the original method is broken down into 10 steps. From the original method, steps 3-6 and 9-10 are modified. We begin by summarizing these steps as they apply to our optimization-based MPM integrator.

1. **Rasterize particle data to the grid.** First, mass and momentum are transferred from particles to the grid using $m_{\mathbf{i}}^n = \sum_p m_p w_{\mathbf{i}p}^n$ and $m_{\mathbf{i}}^n \mathbf{v}_{\mathbf{i}}^n = \sum_p \mathbf{v}_p^n m_p w_{\mathbf{i}p}^n$. Velocity is then obtained by division using $\mathbf{v}_{\mathbf{i}}^n = m_{\mathbf{i}}^n \mathbf{v}_{\mathbf{i}}^n / m_{\mathbf{i}}^n$. Transferring velocity in this way conserves momentum.

2. **Compute particle volumes.** *First time step only.* Our force discretization requires a notion of a particle's volume in the initial configuration. Since cells have a well-defined notion of volume and mass, we can estimate a cell's density as $\rho_{\mathbf{i}}^0 = m_{\mathbf{i}}^0 / h^3$ and interpolate it back to the particle as $\rho_p^0 = \sum_{\mathbf{i}} \rho_{\mathbf{i}}^0 w_{\mathbf{i}p}^0$. Finally, we can define a particle's volume to be $V_p^0 = m_p / \rho_p^0$. Though rather indirect, this approach automatically provides an estimate of the amount of volume that can be attributed to individual particles.

3. **Solve the optimization problem.** Minimize the objective (2.2) using the methods of Section 2.2. This produces a new velocity estimate $\mathbf{v}_{\mathbf{i}}^{n+1}$ on the grid. This step replaces steps 3-6 of the original method.

4. **Update deformation gradient.** The deformation gradient for each particle is updated as $\mathbf{F}_p^{n+1} = (\mathbf{I} + \Delta t \nabla \mathbf{v}_p^{n+1}) \mathbf{F}_p^n$, where we have computed $\nabla \mathbf{v}_p^{n+1} = \sum_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^{n+1} (\nabla w_{\mathbf{i}p}^n)^T$. Note that this involves updates for the elastic and plastic parts of $\mathbf{F}$. See [SSC13] for details, as they are unchanged.

5. **Update particle velocities.** Our new particle velocities are $\mathbf{v}_p^{n+1} = (1 - \alpha) \mathbf{v}_{\mathrm{PIC}p}^{n+1} + \alpha \mathbf{v}_{\mathrm{FLIP}p}^{n+1}$, where the PIC part is $\mathbf{v}_{\mathrm{PIC}p}^{n+1} = \sum_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^{n+1} w_{\mathbf{i}p}^n$ and the FLIP part is $\mathbf{v}_{\mathrm{FLIP}p}^{n+1} = \mathbf{v}_p^n + \sum_{\mathbf{i}} (\mathbf{v}_{\mathbf{i}}^{n+1} - \mathbf{v}_{\mathbf{i}}^n) w_{\mathbf{i}p}^n$. We typically used $\alpha = 0.95$.

6. **Update particle positions.** Particle positions are updated using $\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \mathbf{v}(\mathbf{x}_p^{n+1})$ as described in Section 2.5.3. This step replaces steps 9-10 of the original method.

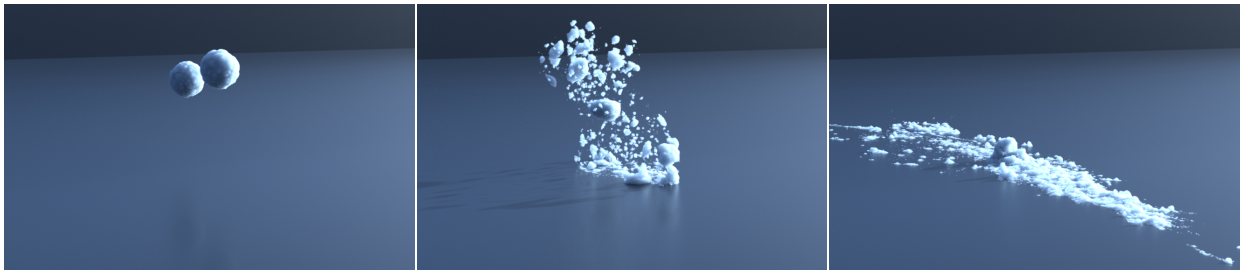


Figure 2.13: *The extension of our method to [SSC13] is robust to large deformation and collisions scenarios. Here we demonstrate this for with two snowballs that smash into each other and fall to the ground.*

2.5.2 Optimization formulation

The primary modification that we propose is to use the optimization framework in place of the original solver. For this, we must formulate their update in terms of an optimization objective E. The original formulation defined the potential energy $\Phi(\mathbf{x_i})$ conceptually in terms of the grid node locations $\mathbf{x_i}$. Here we use the index $\mathbf{i}$ to refer to grid node indices. Their grid is a fixed Cartesian grid and never moves, and they solve for $\mathbf{v}_\mathbf{i}^{n+1}$. We will follow the same conceptual formulation here. This leads to the objective

$$E(\mathbf{v_i}) = \sum_\mathbf{i} \frac{1}{2} m_\mathbf{i} \|\mathbf{v_i} - \mathbf{v}_\mathbf{i}^n\|^2 + \Phi(\mathbf{x}_\mathbf{i}^n + \Delta t \mathbf{v_i}), \tag{2.2}$$

where $m_\mathbf{i}$ is the mass assigned to grid index $\mathbf{i}$. Our final $\mathbf{v}_\mathbf{i}^{n+1}$ is computed so that $E(\mathbf{v}_\mathbf{i}^{n+1})$ is minimized. We solve this minimization problem as in Section 2.2. Note that we apply plasticity explicitly as in the original formulation.

Using larger time steps causes our linear systems to become slower to solve. In the case of MPM, we found it beneficial to use the diagonal preconditioner

$$\mathbf{L_{ii}} = \sum_p \text{diag}(m_p w_{\mathbf{i}p} \mathbf{I} + \Delta t^2 V_p^0 \mathbf{H}),$$

Figure	Ours?	$\frac{\text{Steps}}{\text{frame}}$	$\frac{\text{Time}}{\text{frame}}$ (s)	# dofs	$\frac{\text{Solves}}{\text{step}}$
2.6	Y	5	200	984k	2.2
1.1 mid	Y	1	0.51	18.5k	2.8
1.1 rt	N	1/3	8.7/1.1	18.5k	15/0.7
1.2 top	Y	1	0.52	18.5k	2.9
1.2 bot	N	1/5	3.8/1.3	18.5k	6.6/0.6
2.8 top	Y	1	4.25	28.0k	8.1
2.8 bot	N	1/6	33/7.3	28.0k	26/0.8
2.4	Y	5	1.13	7.9k	2.1
2.3 top	Y	1	68.0^{+}	824k	12.3
2.3 lt	Y	1	1470^{+}	824k	236.8
2.3 rt	Y	1	667^{+}	824k	109.6
2.5 top	Y	1	43.1^{+}	824k	10.7
2.5 lt	Y	1	831^{+}	824k	155.9
2.5 rt	Y	1	444^{+}	824k	88.8
2.7 top	Y	1	0.42	14.0k	3.8
2.7 bot	N	1^{*}	1.13	14.0k	9.8
2.9	Y	1	0.45	7.9k	8.6
2.11	Y	1	46.1	73.8k	34.7
2.10	Y	1	17.1	138k	6.9

Table 2.1: *Optimization integrator performance. Time step sizes and average running times for the examples in the paper. The last column shows the average number of linear solves per time step. Each of the Newton's method examples fails to converge at the frame rate. For fairer comparison, timing information for all but the one marked * is shown at the frame rate and the stable time step size. The stress tests marked $^{+}$ spend the majority of their time on the first frame or two due to the difficult initial state.*

where

$$\mathbf{H} = (\lambda_p + \mu_p)\nabla w_{\mathbf{i}p}\nabla w_{\mathbf{i}p}^T + \mu_p \nabla w_{\mathbf{i}p}^T \nabla w_{\mathbf{i}p}\mathbf{I}.$$

This preconditioner approximates the diagonal of the stiffness matrix at the rest configuration. This works well since snow is unable to deform much without hardening or fracturing. We use an approximation to the diagonal, rather than the exact diagonal, because we never explicitly form the matrix. This approximation suffices for preconditioning and is more efficient.

The original method performed solid body collisions while computing new grid velocities.

We treat body collisions using constraints in our optimization problem. We assume sticking collisions and let $P(\mathbf{v_i}) = \mathbf{0}$ for all grid nodes $\mathbf{i}$ that lie inside a collision object. Note that we do not permit separation during optimization, though separation may occur during other steps in the algorithm.

2.5.3 Particle position update

One of the difficulties with running the method of [SSC13] with larger time steps is the particle-based solid body collisions. They were needed under the old formulation to prevent settling into the ground, but at the same time they cause bunching of particles at collision objects. These problems are exacerbated at larger time steps, and another approach is required. Instead, we show that altering the way we update particle positions can avoid the need for a separate particle collision step.

For each particle position $\mathbf{x}_p$ we solve the backward Euler update equation

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \mathbf{v}(\mathbf{x}_p^{n+1}) \qquad \mathbf{v}(\mathbf{x}_p^{n+1}) = \sum_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^{n+1} N_{\mathbf{i}}^h(\mathbf{x}_p),$$

where $\mathbf{v}(\mathbf{x}_p^{n+1})$ is the interpolated grid velocity at the particle location $\mathbf{x}_p^{n+1}$. These updates are independent per particle and so are relatively inexpensive. A solution to this backward Euler equation always exists nearby provided a suitable CFL condition is respected (no particle moves more than Δx in a time step). Note that pure PIC velocities are used in the particle position updates. While a combination of FLIP/PIC is still stored on particles (to avoid excessive dissipation in subsequent transfer to grid), PIC velocities for position updates lead to more stable behavior.

The motivation for our modification can be best understood in the case of sticking collisions. Inside a collision object, we will have $\mathbf{v}_{\mathbf{i}}^{n+1} = \mathbf{0}$ due to the collision constraints imposed during optimization. If we then assume that we will interpolate $\mathbf{v}(\mathbf{x}_p^{n+1}) = \mathbf{0}$ here, then we can see from $\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \mathbf{v}(\mathbf{x}_p^{n+1})$ that $\mathbf{x}_p^{n+1} = \mathbf{x}_p^n$. Note that if a particle ends up inside the collision object, then it must have already been there. Thus, it is not possible for particles to penetrate collision objects. In our implementation, $\mathbf{v}(\mathbf{x}_p^{n+1}) = \mathbf{0}$ will only be true if we

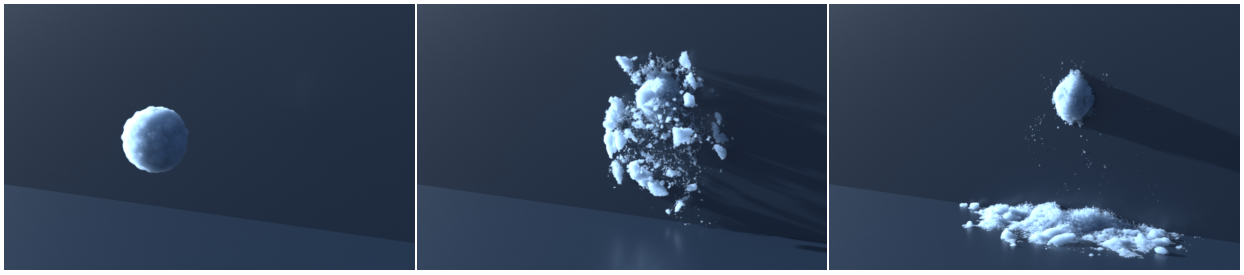


Figure 2.14: *A snowball smashes into a wall and sticks to it.*

are slightly inside collision objects, but in practice this procedure actually stops particles slightly outside collision objects.

We solve this equation with Newton's method. Since Newton's method need not converge, some care is required, though in practice nothing as sophisticated as Section 2.2 is needed. We always use the Newton direction but repeatedly halve the length of the Newton step until the objective $E = \|\mathbf{x}_p^{n+1} - \mathbf{x}_p^n - \Delta t \mathbf{v}(\mathbf{x}_p^{n+1})\|$ no longer increases. (If halving the step size 14 times does not suffice, we take the reduced step anyway.) Typically, only one Newton step is required for convergence. We have never observed this to fail.

We use a quadratic spline rather than the cubic of the original formulation to reduce stencil width and improve the effectiveness of the modified position update. That is, we let

$$N(x) = \begin{cases} \frac{3}{4} - x^2 & |x| < \frac{1}{2} \\ \frac{1}{2}x^2 - \frac{3}{2}|x| + \frac{9}{8} & \frac{1}{2} \le |x| < \frac{3}{2} \\ 0 & |x| \ge \frac{3}{2}. \end{cases}$$

Using a quadratic stencil also has the advantage of being more efficient. We do not use a linear spline since it is not smooth enough for Newton's method to be effective in the particle position update.

Since MPM involves a grid, we limit our time step so that particles do not travel more that one grid spacing per time step. That is, we choose Δt so that $\nu \frac{\Delta x}{\Delta t} \ge \max_p \|\mathbf{v}_p^n\|$ for some $\nu < 1$. We chose $\nu = 0.6$ for our examples. Although the time step restriction is computed

based on $\mathbf{v}_p^n$ rather than $\mathbf{v}_p^{n+1}$, this suffices in practice.

2.6 Results

We begin by demonstrating how robust our solver is by considering the two most difficult constitutive model tests we are aware of: total randomness and total degeneracy. The attributes that make them tough constitutive model tests also make them tough solver tests: high stress, terrible initial guess, tangled configurations, and the need to dissipate massive amounts of unwanted energy. Figure 2.3 shows the recovery of a $65 \times 65 \times 65$ cube (824k dofs) from a randomized initial configuration for three different stiffnesses with $\Delta t = 1/24\,s$. Figure 2.5 repeats the tests with all points starting at the origin. The recovery times vary from about $3\,s$ for the softest to a single time step for the stiffest. We were surprised to find that a single step of backward Euler could untangle a randomized cube, even at high resolution.

Figure 2.4 is a classical torus drop demonstrating that our self collisions are effective at stopping collisions at the torus's hole. Figure 2.9 uses constraints for all collision body collisions and demonstrates that our constraint collisions are effective with concave and convex constraint manifolds. Figure 2.10 demonstrates our method with stiffer deformable bodies with sharp corners. Figure 2.11 demonstrates our constraint collisions are effective for objects with sharp corners. Finally, Figure 2.6 shows a more practical example which uses all three types of collisions: self collisions, constraint collisions (with ground) and penalty collisions (against a bowl defined by a grid-based level set).

2.6.1 MPM results

We demonstrate the advantages of using our optimization integrator by applying it to the MPM snow formulation from [SSC13]. We run three examples using both the original formulation and our modified formulation. We compare with the snowball examples from the original paper. In each case, for our formulation we use the CFL $\nu = 0.6$. Figure 2.14 shows a snowball hitting a wall using sticky collisions, which causes the snow to stick to the wall.

Figure 2.12 shows a dropped snowball hitting the ground with sticky collisions. Figure 2.13 shows two snowballs colliding in mid air with sticky collisions against the ground. On average, we get a speed up of 3.5 times over the original method. These results are tabulated in Table 2.2. Notably, we are able to take significantly larger time steps, however some of the potential gains from this are lost to an increased complexity per time step. Nonetheless, we provide a significant computational savings with minimal modification to the original approach.

2.7 Conclusions

We have demonstrated that backward Euler solved with Newton's method can be made more robust by recasting the resulting system of nonlinear equations as a nonlinear optimization problem so that robust optimization techniques can be employed. The resulting method is extremely robust to large time step sizes, high stress, and tangled configurations.

Runtimes and other performance-related information for all of our sims are provided in Figure 2.1. All Lagrangian simulations were run single-threaded on a $3.1 - 3.5\,GHz$ Xeon core, the MPM simulations were run with 10 threads for 2.13 and 12 threads for 2.12 and 2.14 . Our solver's performance is competitive with a standard Newton solver for those examples where both were run. In general, we take more Newton steps but spend less time on each, and the resulting runtime for typical examples is about the same for the two solvers, though our solver is faster for all of the difficult examples in this paper. Taking a large time step size can actually be slower than taking a smaller one, even with the same solver. For time integrators (like backward Euler) that have a significant amount of damping at large time steps, constitutive models are often tuned to take into account the numerical damping. If the integrator is forced to simulate a portion of a simulation at a smaller time step, the dynamic behavior can change noticeably. Solving with constraints is about the same speed as using penalty collisions.

Note that Figures 2.6 and 2.4 were run with smaller time steps sizes to avoid collision artifacts. This indicates that a self-collision scheme that is more tolerant of large time steps

Figure	Average Δt (s)		Time/frame (s)		Speedup	Grid	# of
	Ours	Orig	Ours	Orig	factor	size	particles
2.12	3.4×10^{-3}	4.4×10^{-4}	59	184	3.1	$600 \times 300 \times 600$	2.8×10^5
2.14	1.5×10^{-3}	1.4×10^{-4}	85	431	5.1	$200 \times 240 \times 600$	2.8×10^5
2.13	1.6×10^{-3}	1.7×10^{-4}	288	780	2.7	$800 \times 300 \times 800$	5.6×10^5

Table 2.2: *Performance comparison of our modified MPM snow formulation ("Ours") with the original formulation ("Orig").*

is required. The scheme does not have problems with collisions between different objects at the frame rate as long as they are not too thin. Continuous collision detection could perhaps be used. We leave both of these problems for future work.

The current method has a couple disadvantages compared with current techniques. It requires a potential energy to exist (which is how most constitutive models are defined anyway) and is sensitive to discontinuities in this energy. The method also occasionally fails to make progress due to valley shaped kinks in our collision processing. In practice, this only occurs when the system is already fairly close to a solution, since otherwise any energy kinks are overwhelmed by the strong gradients in the objective. From a practical perspective, this means this sort of breakdown can be dealt with by simply ignoring it. This does, however, prevent the method from being absolutely robust. We leave this weakness to be addressed in future work.

Our method was derived and implemented on top of a backward Euler integrator, which is known for being very stable but quite damped. The nonlinear system of equations for other A-stable integrators such as trapezoid rule and BDF-2 can also be readily converted into minimization form and solved similarly. Being second order schemes, their use would reduce damping at large time steps, though trapezoid rule's oscillatory properties should be taken into account.

2.8 Acknowledgments

We would like to acknowledge Shunsuke Saito and Yuwei Jiang for their suggestions regarding optimization. All authors were partially supported by NSF (CCF-1422795), DOE

(09-LR-04-116741-BERA), ONR (N000140310071, N000141010730, N000141210834) and Intel STCVisual Computing Grant (20112360).

CHAPTER 3

Viscoelastic Materials

3.1 Governing equations

The governing equations arise from basic conservation of mass and momentum as

$$\frac{D}{Dt}\rho + \rho\nabla\cdot\mathbf{v} = 0, \quad \rho\frac{D}{Dt}\mathbf{v} = \nabla\cdot\boldsymbol{\sigma} + \rho\mathbf{g} \tag{3.1}$$

where ρ is the mass density, $\mathbf{v}$ is the velocity, $\boldsymbol{\sigma}$ is the Cauchy stress and $\mathbf{g}$ is gravitational acceleration. As is commonly done with viscoelastic complex fluids, we write the Cauchy stress as $\boldsymbol{\sigma} = \boldsymbol{\sigma}^N + \boldsymbol{\sigma}^E$ where $\boldsymbol{\sigma}^N = \frac{\mu^N}{2}\left(\frac{\partial\mathbf{v}}{\partial\mathbf{x}} + \frac{\partial\mathbf{v}}{\partial\mathbf{x}}^T\right)$ is the viscous Newtonian component and $\boldsymbol{\sigma}^E$ is the elastic component. We express the constitutive behavior through the elastic component of the left Cauchy Green strain. Specifically, the deformation gradient of the flow $\mathbf{F}$ can be decomposed as a product of elastic and plastic deformation as $\mathbf{F} = \mathbf{F}^E\mathbf{F}^P$ and the elastic left Cauchy Green strain is $\mathbf{b}^E = \mathbf{F}^E(\mathbf{F}^E)^T$ [BW97]. With this convention, we can define the elastic portion of the Cauchy stress via the stored elastic potential $\psi(\mathbf{b}^E)$ as $\boldsymbol{\sigma}^E = \frac{2}{J}\frac{\partial\psi}{\partial\mathbf{b}^E}\mathbf{b}^E$.

3.1.1 Left Cauchy-Green strain plasticity and the upper convected derivative

We can define the plastic flow using the temporal evolution of the elastic right Cauchy Green strain as in [BW97]. Rewriting $\mathbf{F}^E = \mathbf{F}(\mathbf{F}^P)^{-1}$, $\mathbf{b}^E = \mathbf{F}(\mathbf{C}^P)^{-1}\mathbf{F}^T$ where $\mathbf{C}^P = (\mathbf{F}^P)^T\mathbf{F}^P$ is the right plastic Cauchy Green strain. The Eulerian form of the temporal evolution is then obtained by taking the material derivative of $\mathbf{b}^E$ to get

$$\frac{D\mathbf{b}^E}{Dt} = \frac{D\mathbf{F}}{Dt}(\mathbf{C}^P)^{-1}\mathbf{F}^T + \mathbf{F}(\mathbf{C}^P)^{-1}\frac{D\mathbf{F}}{Dt}^T + \mathbf{F}\frac{D}{Dt}[(\mathbf{C}^P)^{-1}]\mathbf{F}^T. \tag{3.2}$$

With this view, the plastic flow is defined via $\frac{D}{Dt}[(\mathbf{C}^P)^{-1}]$. Combining this with $\frac{D}{Dt}\mathbf{F} = \frac{\partial \mathbf{v}}{\partial \mathbf{x}}\mathbf{F}$ (see e.g. [BW97]), the previous equation can be rewritten as

$$\frac{D\mathbf{b}^E}{Dt} = \frac{\partial \mathbf{v}}{\partial \mathbf{x}}\mathbf{b}^E + \mathbf{b}^E\frac{\partial \mathbf{v}}{\partial \mathbf{x}}^T + \mathbf{g}(\mathbf{b}^E) \tag{3.3}$$

where $\mathbf{g}(\mathbf{b}^E) = \mathbf{F}\frac{D}{Dt}[(\mathbf{C}^P)^{-1}]\mathbf{F}^T$ is used to describe the plastic flow rate. This equation is often abbreviated as

$$\overset{\triangledown}{\mathbf{b}}^E = \mathbf{g}(\mathbf{b}^E). \tag{3.4}$$

Here, the operator $\overset{\triangledown}{\mathbf{b}}^E$ (often referred to as the upper convected derivative) is defined to be $\overset{\triangledown}{\mathbf{b}}^E \equiv \frac{D}{Dt}\mathbf{b}^E - \frac{\partial \mathbf{v}}{\partial \mathbf{x}}\mathbf{b}^E - \mathbf{b}^E\frac{\partial \mathbf{v}}{\partial \mathbf{x}}^T$ (see e.g. [Lar99]).

3.1.2 Von Mises plasticity

The Von Mises model [BW97] achieves plasticity through the rate $\mathbf{g}(\mathbf{b}^E) = -2\dot{\gamma}\delta\frac{\partial f(\tau)}{\partial \tau}\mathbf{b}^E$, where τ is the Kirchhoff stress, $\dot{\gamma}$ is the plastic multiplier, $f(\tau)$ is the Von Mises yield condition, and $\delta = 1$ if $f(\tau) \geq 0$, $\delta = 0$ otherwise. However, this is relatively difficult to discretize given the conditional nature of the function. It is often more straightforward to just work directly with $\mathbf{F}^E$ and $\mathbf{F}^P$ in that case (see e.g [SSC13]), however Yue et al [YSB15] do discretize this directly.

3.1.3 Oldroyd-B plasticity

The Oldroyd-B model [Lar99, TFS08] can be see as an alternative definition of $\mathbf{g}(\mathbf{b}^E) = \frac{1}{Wi}(\mathbf{I} - \mathbf{b}^E)$. Combining this with $\mathbf{g}(\mathbf{b}^E) = \mathbf{F}\frac{D}{Dt}[(\mathbf{C}^P)^{-1}]\mathbf{F}^T$ shows that the plastic flow of this model is $\frac{D}{Dt}[(\mathbf{C}^P)^{-1}] = \frac{1}{Wi}(\mathbf{C}^{-1} - (\mathbf{C}^P)^{-1})$ where $\mathbf{C} = \mathbf{F}^T\mathbf{F}$ is the right Cauchy Green strain. This expression for $\mathbf{g}(\mathbf{b}^E)$ is very simple in comparison with that of Von Mises. This simplicity allows for a much easier treatment of temporal discretization needed for implicit time stepping. Specifically, we show in Section 3.2 that this simple definition of $\mathbf{g}(\mathbf{b}^E)$ facilitates the implicit description of the plastic flow in terms of discrete grid node velocities. We can see, both from the $\frac{1}{Wi}(\mathbf{I} - \mathbf{b}^E)$ and $\frac{D}{Dt}[(\mathbf{C}^P)^{-1}] = \frac{1}{Wi}(\mathbf{C}^{-1} - (\mathbf{C}^P)^{-1})$ terms that the

plasticity achieves a strong damping of the elastic component of the stress. The severity of this damping is inversely proportionate to the Weissenberg number Wi. That is, the smaller the Weissenberg number, the faster the elastic strain is damped to the identity, thus releasing elastic potential and associated resistance to deformation. Thus, the Weissenberg number directly controls the amount of the plasticity.

3.1.4 Volume preserving plasticity

The plastic flow in the Oldroyd model will not generally be volume preserving. Since many plastic flows, including those of foams, exhibit this behavior we provide a modification to the standard Oldroyd model that will satisfy this. If we define $\mathbf{b}^E_{OB}$ to obey $\overset{\triangledown}{\mathbf{b}}{}^E_{OB} = \frac{1}{Wi}(\mathbf{I} - \mathbf{b}^E_{OB})$, then we define a new elastic left Cauchy Green strain as

$$\mathbf{b}^E \equiv \left(\frac{J}{J_{OB}}\right)^{\frac{2}{3}} \mathbf{b}^E_{OB}, \tag{3.5}$$

where $J = \det(\mathbf{F})$ and $J_{OB} = \sqrt{\det(\mathbf{b}^E_{OB})}$. Using this definition, $\det(\mathbf{b}^E) = J^2$ and since by definition $\det(\mathbf{b}^E) = \det(\mathbf{F}^E)^2$ and $J = \det(\mathbf{F}^E)\det(\mathbf{F}^P)$ we see that it must be true that $\det(\mathbf{F}^P) = 1$, and thus the plastic flow is volume preserving.

3.1.5 Modified plastic flow

This modification to the Oldroyd plasticity obeys

$$\overset{\triangledown}{\mathbf{b}}{}^E = \frac{D}{Dt}\left(\left(\frac{J}{J_{OB}}\right)^{\frac{2}{3}}\right)\mathbf{b}^E_{OB} + \frac{1}{Wi}\left(\frac{J}{J_{OB}}\right)^{\frac{2}{3}}\left(\mathbf{I} - \mathbf{b}^E_{OB}\right) \tag{3.6}$$

which has the plastic flow

$$\begin{aligned}\frac{D}{Dt}[(\mathbf{C}^P)^{-1}] = \frac{D}{Dt}\left(\left(\frac{J}{J_{OB}}\right)^{\frac{2}{3}}\right)(\mathbf{C}^P_{OB})^{-1} + \\ \frac{1}{Wi}\left(\frac{J}{J_{OB}}\right)^{\frac{2}{3}}\left(\mathbf{C}^{-1} - (\mathbf{C}^P_{OB})^{-1}\right).\end{aligned} \tag{3.7}$$

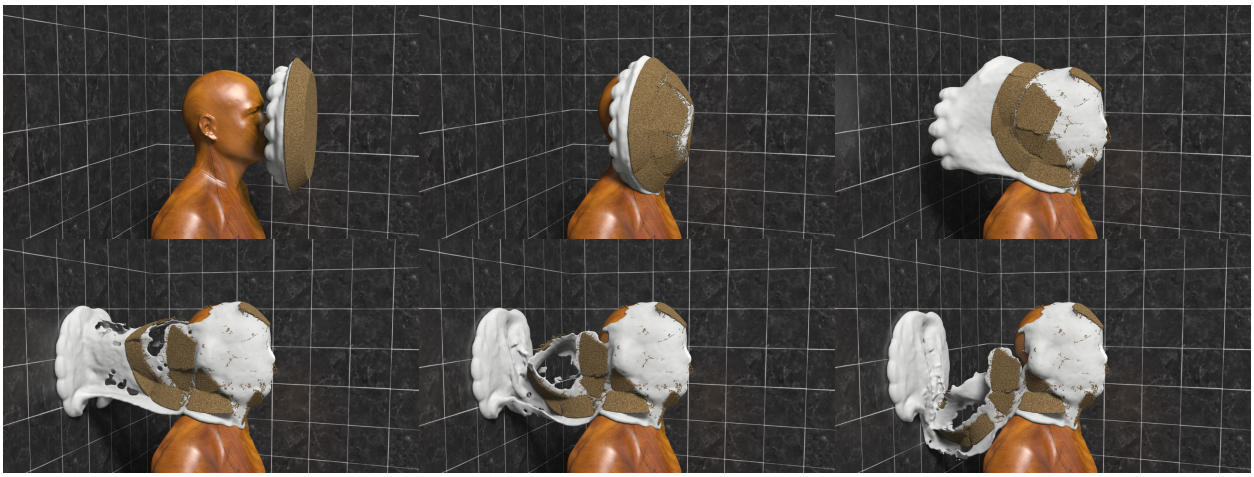


Figure 3.1: *A pie with a stiff crust and soft whipped cream is thrown at a mannequin.*

We do not need to solve for $\mathbf{b}^E$ using the definition of its plastic flow. In practice, we solve for the comparatively simple $\mathbf{b}^E_{OB}$ and then obtain the elastic stress as $\mathbf{b}^E = \left(\frac{J}{J_{OB}}\right)^{\frac{2}{3}}\mathbf{b}^E_{OB}$. We only provide this derivation here to show that there is a plastic flow associated with this definition of the elastic strain.

3.1.6 Elasticity

We define constitutive behavior through the compressible Neo-Hookean elastic potential energy density as

$$\psi(\mathbf{b}^E) = \frac{\mu}{2}(\mathrm{tr}(\mathbf{b}^E) - 3) - \mu\ln(J) + \frac{\lambda}{2}(J-1)^2 \tag{3.8}$$

with associated Cauchy stress

$$\boldsymbol{\sigma}^E = \frac{\mu}{J}(\mathbf{b}^E - \mathbf{I}) + \lambda(J-1)\mathbf{I}. \tag{3.9}$$

3.2 Material point method

We closely follow the algorithm from [SSC13]. The only difference is in the discrete Eulerian grid node forces and force derivatives. All steps in the algorithm not related to the update of grid node velocities are the same; we simply change the nature of stress-based forces. In this section, we describe how to modify the potential-based definition of these forces to discretize

our new governing equations. We refer the reader to [SSC13] for all other steps in the MPM time stepping algorithm.

Using the notation from [SSC13], we denote position, velocity and deformation gradient of particle p at time t^n as $\mathbf{x}_p^n$, $\mathbf{v}_p^n$ and $\mathbf{F}_p^n$ respectively. Eulerian grid node locations are denoted as $\mathbf{x_i}$ where $\mathbf{i} = (i, j, k)$ is the grid node index. The weights at time t^n are $w_{\mathbf{i}p}^n = N_{\mathbf{i}}(\mathbf{x}_p^n)$, where $N_{\mathbf{i}}(\mathbf{x})$ is the interpolation function associated with grid node $\mathbf{i}$ and the weight gradients are $\nabla w_{\mathbf{i}p}^n = \nabla N_{\mathbf{i}}(\mathbf{x}_p^n)$. As in [SSC13], we define the forces on the Eulerian grid nodes as the derivative of an energy with respect to grid node locations. We do not actually move grid nodes, but we consider their movement to define grid node velocities $\mathbf{v_i}$ as $\hat{\mathbf{x}}_\mathbf{i} = \mathbf{x_i} + \Delta t \mathbf{v_i}$. Using $\hat{\mathbf{x}}$ to denote the vector of all grid nodes, we define the potential

$$\Phi(\hat{\mathbf{x}}) = \sum_p \left(\Phi^E(\hat{\mathbf{x}}) V_p^0 + \Phi^N(\hat{\mathbf{x}}) V_p^n \right) \tag{3.10}$$

where $\Phi^E(\hat{\mathbf{x}})$ is the elastoplastic component of the potential energy density $\Phi^E(\hat{\mathbf{x}}) = \psi\big(\hat{\mathbf{b}}^E(\hat{\mathbf{x}})\big)$ and $\Phi^N(\hat{\mathbf{x}})$ is the Newtonian viscous potential energy density

$$\Phi^N(\hat{\mathbf{x}}) = \mu^N \hat{\boldsymbol{\epsilon}}_p(\hat{\mathbf{x}}) : \hat{\boldsymbol{\epsilon}}_p(\hat{\mathbf{x}}) = \sum_{i,j} \mu^N \hat{\epsilon}_{p_{ij}}(\hat{\mathbf{x}}) \hat{\epsilon}_{p_{ij}}(\hat{\mathbf{x}}). \tag{3.11}$$

Here $\hat{\boldsymbol{\epsilon}}_p(\hat{\mathbf{x}}) = \frac{1}{2}\big(\nabla\hat{\mathbf{v}}(\hat{\mathbf{x}}) + (\nabla\hat{\mathbf{v}}(\hat{\mathbf{x}}))^T\big)$ is the strain rate at $\mathbf{x}_p^n$ induced by the grid node motion defined by $\hat{\mathbf{x}}$ over the time step and $\nabla\hat{\mathbf{v}}(\hat{\mathbf{x}}) = \sum_\mathbf{i} \frac{\hat{\mathbf{x}}_\mathbf{i} - \mathbf{x_i}}{\Delta t} (\nabla w_{\mathbf{i}p}^n)^T$. As in [SSC13], V_p^0 is the volume of the material originally occupied by the particle p. However, for the viscous Newtonian potential, we are approximating an integral over the time t^n configuration of the material so we have $V_p^n = \det(\mathbf{F}_p^n) V_p^0$.

As in [SSC13], we store a deformation gradient $\mathbf{F}_p^n$ on each particle and update it using

$$\hat{\mathbf{F}}(\hat{\mathbf{x}}) = (\mathbf{I} + \Delta t \nabla\hat{\mathbf{v}}(\hat{\mathbf{x}})) \mathbf{F}_p^n. \tag{3.12}$$

We use this to define $\hat{J}_p(\hat{\mathbf{x}}) = \det(\hat{\mathbf{F}}(\hat{\mathbf{x}}))$ in the definition of

$$\hat{\mathbf{b}}^E(\hat{\mathbf{x}}) = \left(\frac{\hat{J}_p(\hat{\mathbf{x}})^2}{\det(\hat{\mathbf{b}}^E_{OB_p}(\hat{\mathbf{x}}))}\right)^{\frac{1}{3}} \hat{\mathbf{b}}^E_{OB_p}(\hat{\mathbf{x}}). \tag{3.13}$$

Similar to the treatment in Equation 3.12, we store $\mathbf{b}^{E^n}_{OB_p}$ on each particle and discretize the upper convected derivative terms in the evolution equation for $\mathbf{b}^E_{OB}$ to get

$$\begin{aligned}\hat{\mathbf{b}}^E_{OB_p}(\hat{\mathbf{x}}) =& \Delta t \nabla\hat{\mathbf{v}}(\hat{\mathbf{x}})\mathbf{b}^{E^n}_{OB_p} + \Delta t \mathbf{b}^{E^n}_{OB_p}(\nabla\hat{\mathbf{v}}(\hat{\mathbf{x}}))^T \\ &+ \frac{\Delta t}{Wi}\mathbf{I} + \left(1 - \frac{\Delta t}{Wi}\right)\mathbf{b}^{E^n}_{OB_p}\end{aligned} \tag{3.14}$$

The force on the grid nodes is defined as $\mathbf{f}(\hat{\mathbf{x}}) = -\frac{\partial\Phi}{\partial\hat{\mathbf{x}}}(\hat{\mathbf{x}})$ and it is used in the implicit update of grid velocities $\mathbf{v}_\mathbf{i}^{n+1}$ exactly as in [SSC13]. We work out these derivatives as well as the $\frac{\partial\mathbf{f}}{\partial\hat{\mathbf{x}}}(\hat{\mathbf{x}})$ in (§3.5).

3.3 Results

In Figure 1.3, a sponge is twisted with top and bottom fixed by Dirichlet boundary conditions. Dynamic fracture and self collision are naturally handled. In Figure 1.6, the top and bottom of a sponge are held in place as we shoot it with a kinematic bullet. The animation is in slow motion to show the detailed material response after the impact. In Figure 3.3, we simulate a stream of shaving foam hitting the ground, and compare it with real world footage. Our method captures the S-shaped buckling and merging behaviors. It also exhibits similar elasto-plastic responses. In Figure 3.2, we simulate toothpaste falling onto a toothbrush. Unlike the shaving foam, Newtonian viscosity dominates material behavior. Figure 1.4 shows a simulation of manufacturing Viennetta ice cream. It captures the characteristic folding behavior. In Figure 3.1, we model a pie and throw it at a mannequin. The fracture pattern of the crust is prescored with weak MPM particles. The cream exhibits detailed splitting and merging behavior. For the particle-grid transfers, we used the affine Particle-In-Cell (APIC) method from [JSS15]. We found that using APIC greatly reduced positional artifacts of the

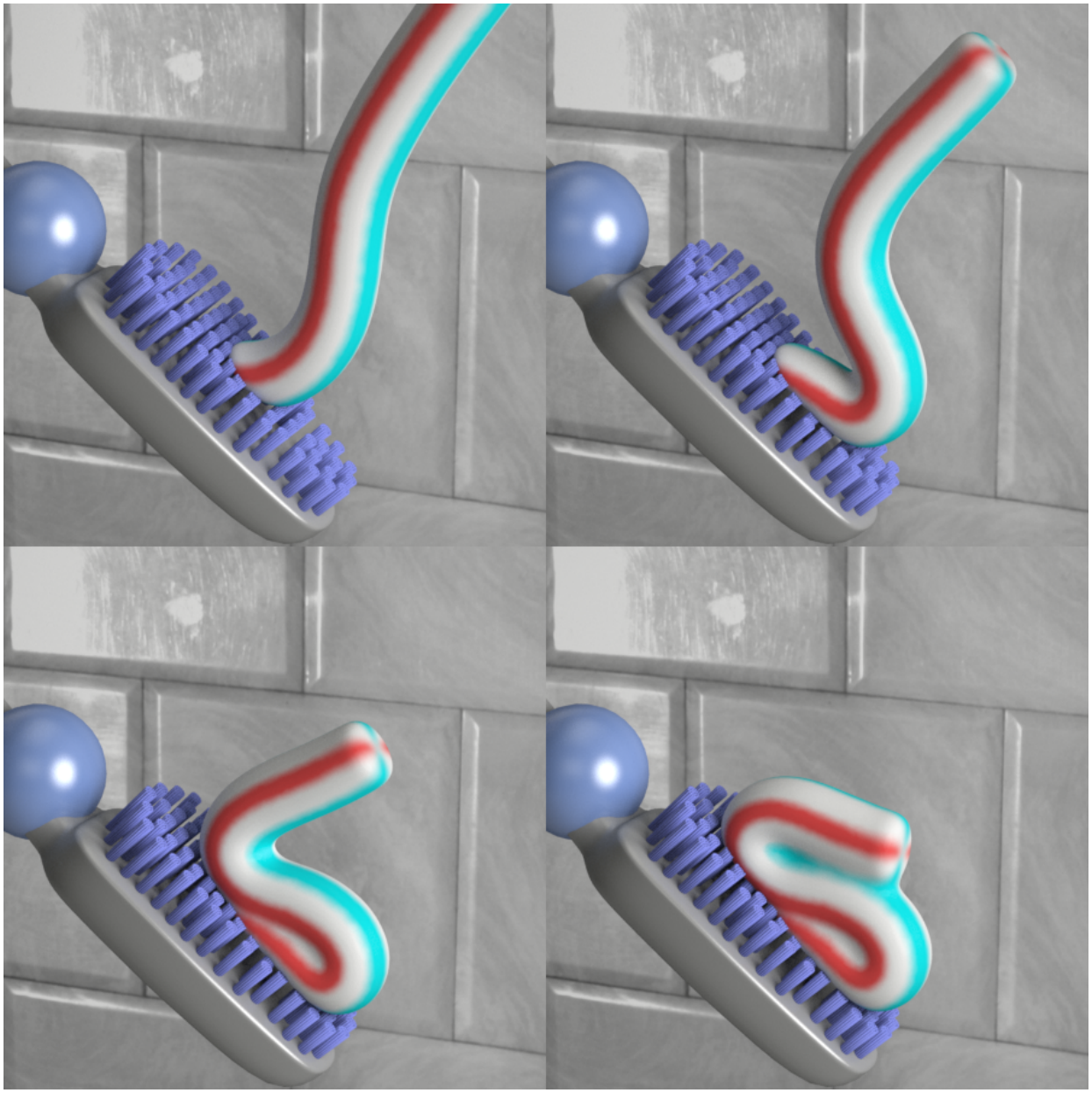


Figure 3.2: *A simulation of toothpaste. Unlike the shaving foam, Newtonian viscosity dominates material behavior.*

pie particles. We do not perform any explicit particle resampling because self-collision and topology change are naturally handled by MPM.

The material parameters used in our examples are given in Table 3.1. The simulation times are shown in Table 3.3. All simulations were performed on Intel Xeon machines. All renderings were done with Mantra in Houdini. For foam, toothpaste, and Viennetta ice cream, surfaces were reconstructed with OpenVDB [Mus14] and rendered with subsurface

	ρ	μ	λ	μ^N	Wi
Twisting sponge	2	3.6×10^2	1.4×10^3	0	50
Shooting sponge	1	3.6×10^2	1.4×10^3	0	50
Shaving foam	0.2	5	50	1×10^{-4}	0.5
Toothpaste	1	0.839	8.39	1×10^{-1}	0.4
Viennetta ice cream	1	1	10	5×10^{-5}	0.1
Pie cream	0.2	5	50	1×10^{-7}	1×10^{-4}
Pie crust	0.5	5×10^5	4×10^6	1×10^{-8}	1×10^{30}
Pie crust scored	0.5	5	10	1×10^{-5}	1

Table 3.1: *Viscoelastic material parameters.*

scattering. The sponges were rendered as a density field.

3.4 Discussions

We found that using a Jacobi preconditioner greatly reduced simulation run times. For example, in the shooting sponge test (Figure 1.6), the Jacobi preconditioner reduces the number of CG iterations by a factor of 6.

While we have used our method successfully in simulating a variety of materials, it has some limitations. Many of these are related to the Oldroyd-B model. For example, unlike the approach in [YSB15], our approach cannot handle shear thickening. Therefore, the model cannot be applied to materials such as oobleck. Our method also does not handle material softening or hardening.

Our update rule of $\mathbf{b}_{OB}^E$ allows for inversion which the constitutive model cannot handle. While $\mathbf{b}_{OB}^E$ should remain positive definite, we have found this to be only partially required. In particular, (3.8) involves the quantity $\text{tr}(\mathbf{b}^E)$, which we must ensure is bounded from below. If $\mathbf{b}^E$ is positive definite, then $\text{tr}(\mathbf{b}^E) > 0$. We also compute $\det(\mathbf{b}_{OB}^E)^{-\frac{1}{3}}$, which is problematic if $\mathbf{b}_{OB}^E$ may become singular. We avoid these problems in practice by taking advantage of the optimization-based integrator from [GSS15]. We add a large penalty to our objective when the determinant or trace of $\mathbf{b}_{OB}^E$ becomes infeasible; the line search in our optimizer then discards these configurations. While bounding the trace and determinant does not enforce definiteness in 3D, this strategy worked well in practice. Not enforcing these produces popping artifacts.

Method	Elastoplastic	Viscosity	No SVD	Implicit	No Remeshing
[BB08]	✗	✓	✓	✓	✓
[WTG09]	✓	✓	✗	○†	✗
[BH11]	✗	✓	✓	✓	○‡
[BUA12]	✗	✓	✓	✓	✗
[SSC13]	✓	✗	✗	✓	✓
[SSJ14]	✓	✗	✗	✓	✓
[YSB15]	✓	✓	✓	✗	✓
Our method	✓	✓	✓	✓	✓

Table 3.2: *Viscoelastic feature comparison.* †*This method is not implicit in elasticity.* ‡*This method requires adaptive refinement of a BCC lattice.*

	Min/Frame	Particle #	Threads	CPU	Δx	Grid Resolution
Twisting sponge	5.3	9.1×10^5	20	3.00GHz	0.0366	245^3
Shooting sponge	2.0	7.2×10^5	16	2.90GHz	0.0402	175^3
Shaving foam	0.93	1.1×10^6	12	3.47GHz	0.0019	257^3
Toothpaste	0.28	2.8×10^5	16	2.90GHz	0.0082	$244 \times 487 \times 244$
Viennetta ice cream	1.11	1.2×10^6	12	2.67GHz	0.0026	$385 \times 96 \times 64$
Pie	23.6	1.3×10^6	12	3.07GHz	0.0024	333^3

Table 3.3: *Viscoelastic simulation performance.*

Acknowledgements

UCLA authors were partially supported by NSF CCF-1422795, ONR (N000141110719, N000141210834), Intel STC-Visual Computing Grant (20112360) as well as a gift from Disney Research. We also thank Matthew Wang for providing his 3D head model for the pie example.

3.5 Derivatives

While the potential energy contains many elements and computing its second derivatives seems like a hopeless task, this is not the case. Breaking the potential energy into small pieces makes the implementation straightforward to implement and debug. We present pseudocode that may be used to compute the potential energy $\Phi = \sum_p \Phi_p$ along with its derivatives, $\frac{\partial \Phi}{\partial \mathbf{x_i}} = \sum_p \Phi_{p,\mathbf{i}}$ and $\frac{\partial^2 \Phi}{\partial \mathbf{x_i} \partial \mathbf{x_i}} = \sum_p \Phi_{p,\mathbf{ij}}$. The following computational steps may be used to compute the potential energy contribution of a particle Φ_p. Note that all of the quantities computed below, except for the final result Φ_p, are intermediate quantities used to break the computation into many parts. Most of them have no particular physical

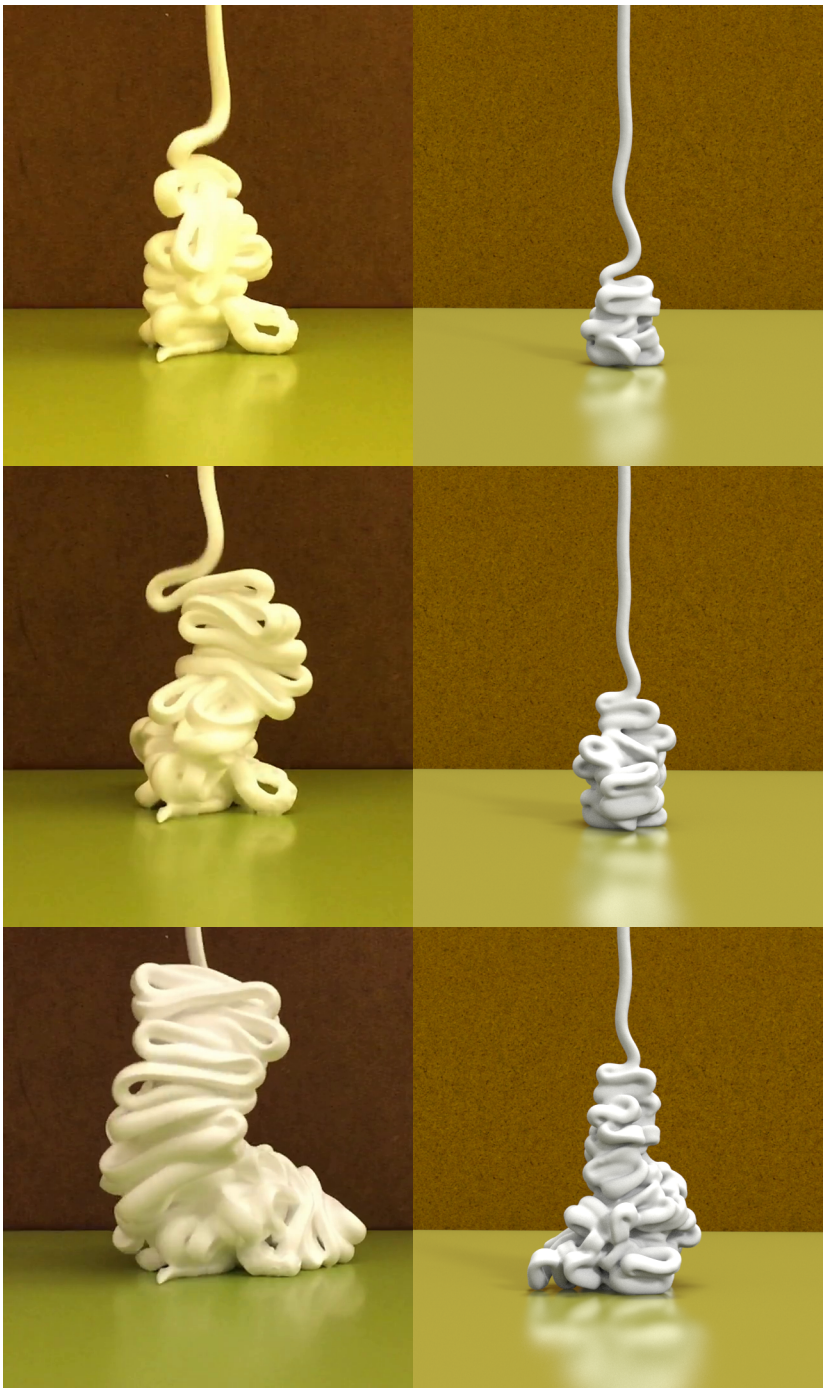


Figure 3.3: *Simulated shaving foam (right) is compared with real world footage (left). The simulation captures the characteristic S-shaped buckling and elastic behavior.*

significance, and most have no particular relationship to similarly named quantities elsewhere

in this manuscript. The bold capitalized quantities are matrices, and the rest are scalars.

$$\mathbf{A}_p \leftarrow \sum_{\mathbf{i}} (\hat{\mathbf{x}}_{\mathbf{i}} - \mathbf{x}_{\mathbf{i}}^n)(\nabla w_{\mathbf{i}p}^n)^T \qquad \mathbf{B}_p \leftarrow \mathbf{A}_p \mathbf{b}_{OBp}^{E^n}$$

$$\mathbf{G}_p \leftarrow \mathbf{b}_{OBp}^{E^n} + \frac{\Delta t}{Wi}\big(\mathbf{I} - \mathbf{b}_{OBp}^{E^n}\big) \qquad \hat{\mathbf{F}}_p \leftarrow (\mathbf{I} + \mathbf{A}_p)\mathbf{F}_p^n$$

$$\mathbf{S}_p \leftarrow \mathbf{G}_p + \mathbf{B}_p + \mathbf{B}_p^T \qquad \mathbf{H}_p \leftarrow \hat{\mathbf{F}}_p^{-1}$$

$$J_p \leftarrow \det(\hat{\mathbf{F}}_p) \qquad a_p \leftarrow \frac{\lambda}{2}(J_p - 1)^2$$

$$q_p \leftarrow \frac{1}{2\Delta t^2}\big(\|\mathbf{A}_p\|_F^2 + \mathbf{A}_p^T : \mathbf{A}_p\big) \qquad b_p \leftarrow \mu \ln(J_p)$$

$$c_p \leftarrow \mu_P^N q_p \det(\mathbf{F}_p^n) \qquad g_p \leftarrow \mathrm{tr}(\mathbf{S}_p)$$

$$\mathbf{K}_p \leftarrow \mathbf{S}_p^{-1} \qquad h_p \leftarrow \det(\mathbf{S}_p)$$

$$k_p \leftarrow h_p^{-\frac{1}{d}} \qquad m_p \leftarrow J_p^{\frac{2}{d}}$$

$$n_p \leftarrow k_p g_p \qquad p_p \leftarrow \frac{\mu}{2} m_p n_p$$

$$\Phi_p \leftarrow V_p(p_p - b_p + a_p + c_p)$$

The next set of routines are for the first derivatives of the quantities above, with the final result being the potential energy derivative for a particle, $\Phi_{p,\mathbf{i}}$. Note that these routines use the quantities computed above. Intermediate quantities of the form $c_{p,\mathbf{i}}$ are related to the intermediates above by $c_{p,\mathbf{i}} = \frac{\partial c_p}{\partial \hat{\mathbf{x}}_{\mathbf{i}}}$, which allows for incremental testing. All quantities computed below are vectors.

$$\overline{b}_{p\mathbf{i}} \leftarrow \mathbf{b}_{OBp}^{E^n} \nabla w_{\mathbf{i}p}^n \qquad \overline{f}_{p\mathbf{i}} \leftarrow (\mathbf{F}_p^n)^T \nabla w_{\mathbf{i}p}^n$$

$$\overline{h}_{p\mathbf{i}} \leftarrow \mathbf{H}_p^T \overline{f}_{p\mathbf{i}} \qquad \overline{k}_{p\mathbf{i}} \leftarrow \mathbf{K}_p^T \overline{b}_{p\mathbf{i}}$$

$$J_{p,\mathbf{i}} \leftarrow J_p \overline{h}_{p\mathbf{i}} \qquad a_{p,\mathbf{i}} \leftarrow \lambda(J_p - 1) J_{p,\mathbf{i}}$$

$$q_{p,\mathbf{i}} \leftarrow \frac{1}{\Delta t^2}(\mathbf{A}_p \nabla w_{\mathbf{i}p}^n + \mathbf{A}_p^T \nabla w_{\mathbf{i}p}^n) \qquad b_{p,\mathbf{i}} \leftarrow \mu \overline{h}_{p\mathbf{i}}$$

$$c_{p,\mathbf{i}} \leftarrow \mu_P^N \det(\mathbf{F}_p^n) q_{p,\mathbf{i}} \qquad g_{p,\mathbf{i}} \leftarrow 2\overline{b}_{p\mathbf{i}}$$

$$k_{p,\mathbf{i}} \leftarrow -\frac{2k_p}{d} \overline{k}_{p\mathbf{i}} \qquad m_{p,\mathbf{i}} \leftarrow \frac{2m_p}{d} \overline{h}_{p\mathbf{i}}$$

$$p_{p,\mathbf{i}} \leftarrow \frac{\mu}{2}(m_{p,\mathbf{i}} n_p + m_p n_{p,\mathbf{i}}) \qquad n_{p,\mathbf{i}} \leftarrow k_{p,\mathbf{i}} g_p + k_p g_{p,\mathbf{i}}$$

$$\Phi_{p,\mathbf{i}} \leftarrow V_p(p_{p,\mathbf{i}} - b_{p,\mathbf{i}} + a_{p,\mathbf{i}} + c_{p,\mathbf{i}})$$

The final set of routines are for second derivatives, with the final result being the potential energy Hessian for a particle, $\Phi_{p,\mathbf{ij}}$. Intermediate quantities of the form $c_{p,\mathbf{ij}}$ are related to the intermediates above by $c_{p,\mathbf{ij}} = \frac{\partial c_{p,\mathbf{i}}}{\partial \hat{\mathbf{x}}_\mathbf{j}}$. All quantities computed below are matrices.

$$
\begin{aligned}
J_{p,\mathbf{ij}} &\leftarrow J_p \overline{h}_{p,\mathbf{i}} \overline{h}_{p,\mathbf{j}}^T - J_p \overline{h}_{p,\mathbf{j}} \overline{h}_{p,\mathbf{i}}^T \\
a_{p,\mathbf{ij}} &\leftarrow \lambda J_{p,\mathbf{i}} J_{p,\mathbf{j}}^T + \lambda(J_p - 1) J_{p,\mathbf{ij}} \\
b_{p,\mathbf{ij}} &\leftarrow -\mu \overline{h}_{p,\mathbf{j}} \overline{h}_{p,\mathbf{i}}^T \\
q_{p,\mathbf{ij}} &\leftarrow \frac{1}{\Delta t^2}((\nabla w_{\mathbf{ip}}^n)^T \nabla w_{,\mathbf{j}} \mathbf{I} + \nabla w_{,\mathbf{j}} (\nabla w_{\mathbf{ip}}^n)^T) \\
c_{p,\mathbf{ij}} &\leftarrow \mu_P^N \det(\mathbf{F}_p^n) q_{p,\mathbf{ij}} \\
k_{p,\mathbf{ij}} &\leftarrow \frac{4k_p}{d^2} \overline{k}_{p,\mathbf{i}} \overline{k}_{p,\mathbf{j}}^T + \frac{2k_p}{d} \overline{k}_{p,\mathbf{j}} \overline{k}_{p,\mathbf{i}}^T + \frac{2k_p}{d} \overline{b}_{p,\mathbf{i}}^T \overline{k}_{p,\mathbf{j}} \mathbf{K}_p \\
m_{p,\mathbf{ij}} &\leftarrow \frac{4m_p}{d^2} \overline{h}_{p,\mathbf{i}} \overline{h}_{p,\mathbf{j}}^T - \frac{2m_p}{d} \overline{h}_{p,\mathbf{j}} \overline{h}_{p,\mathbf{i}}^T \\
n_{p,\mathbf{ij}} &\leftarrow k_{p,\mathbf{ij}} g_p + k_{p,\mathbf{i}} g_{,\mathbf{j}}^T + g_{p,\mathbf{i}} k_{,\mathbf{j}}^T \\
p_{p,\mathbf{ij}} &\leftarrow \frac{\mu}{2}(m_{p,\mathbf{ij}} n_p + m_{p,\mathbf{i}} n_{,\mathbf{j}}^T + n_{p,\mathbf{i}} m_{,\mathbf{j}}^T + n_{p,\mathbf{ij}} m_p) \\
\Phi_{p,\mathbf{ij}} &\leftarrow V_p(p_{p,\mathbf{ij}} - b_{p,\mathbf{ij}} + a_{p,\mathbf{ij}} + c_{p,\mathbf{ij}})
\end{aligned}
$$

CHAPTER 4

Granular Materials

4.1 Background

Conservation laws. We represent the sand as an elastoplastic continuum, whose state can be described at each location by its density $\rho(x, y, z)$ and velocity $\mathbf{v}(x, y, z)$. Our sand experiences internal stress $\boldsymbol{\sigma}$ and gravity $\mathbf{g}$. The motion of the sand satisfies conservation of mass

$$\frac{D\rho}{Dt} + \rho\nabla \cdot \mathbf{v} = 0 \tag{4.1}$$

and conservation of momentum, which can be simplified to the Euler-Lagrange equations,

$$\rho\frac{D\mathbf{v}}{Dt} = \nabla \cdot \boldsymbol{\sigma} + \rho\mathbf{g}. \tag{4.2}$$

Figure 4.1: *Sand falls through the narrow neck of an hourglass, accumulating at the bottom.*

Here, we have used $\frac{D\phi}{Dt} = \frac{\partial \phi}{\partial t} + \mathbf{v} \cdot \nabla \phi$ to denote the material derivative of an arbitrary function $\phi(x, y, z)$. The text of Gonzalez and Stuart [GS08] provides useful background for these equations.

Deformation gradient. The deformation gradient represents how deformed a material is locally. For example, let $\mathbf{x}_1^0$ and $\mathbf{x}_2^0$ be two nearby points embedded in the material (see Figure 4.2) at the beginning of the simulation, and let $\mathbf{x}_1$ and $\mathbf{x}_2$ be the same two points in the current configuration. Then $(\mathbf{x}_2 - \mathbf{x}_1) = \mathbf{F}(\mathbf{x}_2^0 - \mathbf{x}_1^0)$. The deformation gradient $\mathbf{F}$ evolves according to

$$\frac{D\mathbf{F}}{Dt} = (\nabla \mathbf{v})\mathbf{F}. \tag{4.3}$$

Elastic and plastic deformation gradient. We represent plasticity by factoring deformation gradient into elastic and plastic parts as $\mathbf{F} = \mathbf{F}^E \mathbf{F}^P$. The deformation gradient is a measure of how a material has locally rotated and deformed due to its motion. By factoring the deformation gradient in this way, we divide this deformation history into two pieces. The plastic part, $\mathbf{F}^P$, represents the portion of the material's history that has been forgotten. If a metal rod is bent into a coiled spring, the rod *forgets* that it used to be straight; the coiled spring behaves as though it was always coiled (see Figure 4.3). The twisting and bending involved in this operation is stored in $\mathbf{F}^P$. If the spring is compressed slightly, the spring will feel strain (deformation). This is elastic deformation, which is stored in $\mathbf{F}^E$. The spring remembers this deformation. In response, the material exerts stress to try to restore itself to its coiled shape. In this way, we see that only $\mathbf{F}^E$ should be used to compute stress. The full history of the metal rod consists of being bent into a spring shape ($\mathbf{F}^P$) and then being

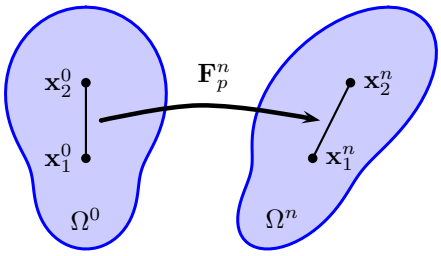


Figure 4.2: *Relationship between deformation and* $\mathbf{F}_p^n$.

compressed ($\mathbf{F}^E$).

Constitutive model. A constitutive model relating the state to the stress is needed. The text of Bonet and Wood [BW08] provides useful background for elastoplastic constitutive modeling. Following Mast el al. [Mas13, MAM14], we use Drucker-Prager elastoplasticity [DP52]. The elastic part of this relation is expressed through the deformation gradient $\mathbf{F}$.

For perfectly hyperelastic materials the constitutive relation is defined through the potential energy, which increases with non-rigid deformation from the initial state. However, in the case of large-strain elastoplasticity, there will be some permanent (or plastic) deformation and the potential will only increase for deformation beyond this state. In this case, the stress in the material is

$$\boldsymbol{\sigma} = \frac{1}{\det(\mathbf{F})} \frac{\partial \psi}{\partial \mathbf{F}^E} \mathbf{F}^{E^T} \tag{4.4}$$

where $\psi(\mathbf{F}^E)$ is the elastic energy density designed to penalize non-rigid $\mathbf{F}^E$ (see Section 4.4.3 for more detailed discussion).

With the Drucker-Prager model, frictional interactions between grains of sand can be expressed in the continuum via a relation between shear and normal stresses. Using a Coulomb friction model, shear stresses resisting sliding motions between grains can only be as large as a constant times the normal stress holding them together. For example, if shear stresses larger than this value are required to maintain a static pile, plastic flow will commence when the limit is reached and the material will move. This constraint defines a

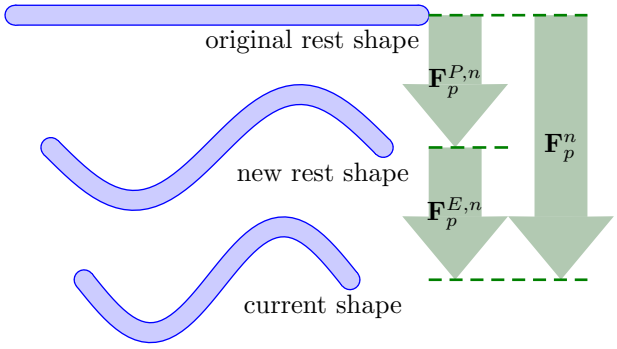


Figure 4.3: *Relationship between* $\mathbf{F}_p^n$, $\mathbf{F}_p^{E,n}$, *and* $\mathbf{F}_p^{P,n}$.

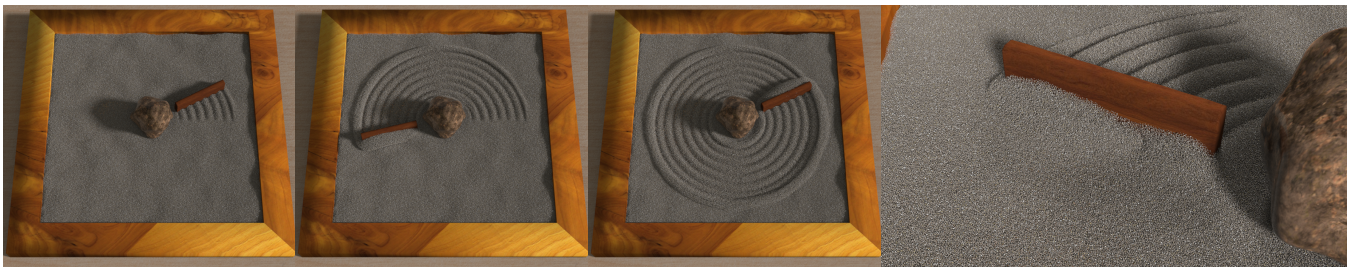


Figure 4.4: *A rake is dragged around a rock, producing a circular pattern in the sand.*

feasible region of stresses, the surface of the feasible region is often referred to as the yield surface. The decomposition of the deformation gradient into elastic and plastic components $\mathbf{F} = \mathbf{F}^E\mathbf{F}^P$ can be viewed as a means of projecting the deformation to satisfy the constraint. Notably, the projection must be designed carefully to ensure for increase of entropy as well as volume preserving plastic flow. We discuss the model in more detail in (§4.5) as well as (§4.14).

Discretization. Traditional approaches for discretization are typically either Eulerian or Lagrangian, which differ by their frame of reference. An Eulerian description computes quantities of interest at fixed locations in space. These methods feature fixed grids. Eulerian methods are ideal for handling collisions and changes in topology, making them a popular choice for fluids.

A Lagrangian description uses quantities that move with the material being described. These methods tend to use moving particles often connected by a mesh. This representation automatically conserves mass, and the mesh provides a straightforward way to determine how deformed the material is. Lagrangian methods are preferred for elastic solids.

Some materials, such as sand, exhibit characteristics of both fluids and solids. Sand can support a load like a solid, but it can also flow like a liquid. For materials like these, there is growing interest in hybrid methods, such as the Material Point Method, which combine aspects of both types of discretization, seeking to obtain some of the benefits of each.

The Material Point Method stores information on Lagrangian particles, but it computes forces using a fixed Eulerian grid. The use of particles makes mass conservation trivial, and it provides a simple means of moving information around. The use of a fixed grid provides auto-

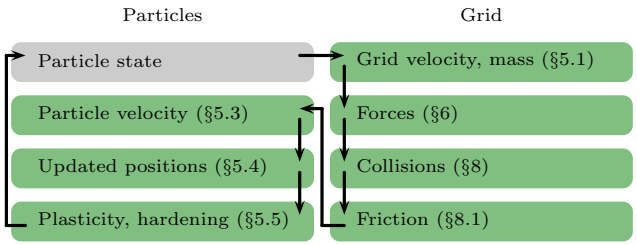


Figure 4.5: *Overview of MPM stages.*

matic handling of topology changes (merging and separating) and collisions between regions of material. Since MPM uses two distinct representations, information must be transferred between them. These transfers play a very important role in the numerical behavior of a hybrid method. Furthermore, to simplify topology changes, MPM does not store connectivity between particles. This avoids the need for complex remeshing, but deformation must now be tracked in an Eulerian way.

In the next section, we outline our discretization steps.

4.2 Overview

Before presenting the algorithm in detail, we first provide an overview of the steps that are involved in the algorithm and the role that they play, which is summarized in Figure 4.5.

1. **Transfer to grid.** Transfer mass and momentum from particles to the grid. Use mass and momentum to compute velocity on the grid. (§4.3.1)

2. **Apply forces.** Compute elastic forces using a deformation gradient that has been projected into the plastic yield surface and apply the forces to the grid velocities. (§4.4)

3. **Grid collisions.** Project grid velocities for collisions against scripted bodies and obstacles, ignoring friction (§4.6). For implicit, this is merged with the force application step (§4.3.6).

4. **Friction.** Compute and apply friction based on the collisions that were resolved. The

velocity before and after this step are retained for use during the transfers. (§4.6.1)

5. **Transfer to particles.** Transfer velocities from grid to particles, being careful to handle friction in a manner that does not lead to inconsistencies. (§4.3.3)

6. **Update particles.** Update remaining particle state, including positions and deformation gradient. (§4.3.4)

7. **Plasticity and hardening.** Project the deformation gradient for plasticity, updating the elastic and plastic parts. Perform hardening, which updates the plastic yield surface. (§4.3.5)

4.3 Algorithm

Notation. It is helpful to establish the conventions for notation (see Table 4.1 for a complete list). Scalars are represented by non-bold Latin or Greek characters (m_p, α_p^n, G_k). Vectors are represented by bold lowercase Latin characters ($\mathbf{v}_p^n$, $\overline{\mathbf{x}}_i^{n+1}$). Matrices are represented by bold uppercase Latin characters or bold Greek characters ($\mathbf{I}$, $\hat{\mathbf{F}}_p^{P,n+1}$, $\boldsymbol{\sigma}$). Derivatives alter this in the usual way, so that ∇G_{ki} is a vector and $(\nabla \mathbf{v})_p$ is a matrix.

Many quantities are indexed with subscripts, which indicate where quantities are stored. Quantities that are stored at grid nodes are indexed with i and particle quantities have the index p. Collision-related quantities have an index k relating them to a particular collision interaction. A quantity may have more than one subscript (w_{ip}^n, ∇G_{ki}). The quantity $\mathbf{F}_p^{n+1}$ represents the quantity corresponding to one index, and $\langle \mathbf{F}_p^{n+1} \rangle$ represents a vector of all such quantities.

Superscript n is used to indicate a quantity near the beginning of the time step, before forces are applied, (m_i^n, $\mathbf{B}_p^n$). Superscript $n+1$ indicates a quantity near the end of the time step, after forces are applied, ($\overline{\mathbf{x}}_i^{n+1}$, $\mathbf{F}_p^{P,n+1}$).

Stars, tildes, and bars are used to distinguish intermediate quantities ($\mathbf{v}_i^\star$, $\tilde{\mathbf{v}}_i^{n+1}$, $\overline{\mathbf{v}}_i^{n+1}$), and some effort is made to group them, but the adornments do not have any intrinsic meaning.

Variable	Where	Type	Meaning
$\mathbf{I}$	-	matrix	identity matrix
Δt	-	scalar	time step size
h	-	scalar	grid resolution
$\frac{D}{Dt}$	-	-	material derivative
$\mathbf{g}$	-	vector	gravity
$\boldsymbol{\sigma}$	-	matrix	Cauchy stress
ρ	-	scalar	density
$\mathbf{v}$	-	vector	velocity
m_p	particles†	scalar	particle mass
V_p^0	particles†	scalar	initial particle volume
α_p^n, α_p^{n+1}	particles†	scalar	yield surface size
q_p^n, q_p^{n+1}	particles†	scalar	hardening state
$\mathbf{B}_p^n$, $\mathbf{B}_p^{n+1}$	particles†	matrix	affine momentum
$\mathbf{F}_p^n$, $\mathbf{F}_p^{n+1}$	particles	matrix	deformation gradient
$\mathbf{F}_p^{E,n}$, $\mathbf{F}_p^{E,n+1}$	particles†	matrix	elastic deformation gradient
$\mathbf{F}_p^{P,n}$, $\mathbf{F}_p^{P,n+1}$	particles†	matrix	plastic deformation gradient
$\mathbf{v}_p^n$, $\mathbf{v}_p^{n+1}$	particles†	vector	particle velocity
$\mathbf{x}_p^n$, $\mathbf{x}_p^{n+1}$	particles†	vector	particle position
$\mathbf{C}_p^n$	particles	matrix	particle velocity derivative (APIC)
$\mathbf{D}_p^n$	particles	matrix	affine inertia tensor (APIC)
$\hat{\mathbf{F}}_p^{n+1}$	particles	matrix	deformation gradient, before plasticity
$\hat{\mathbf{F}}_p^{E,n+1}$	particles	matrix	elastic deformation gradient, before plasticity
$\hat{\mathbf{F}}_p^{P,n+1}$	particles	matrix	plastic deformation gradient, before plasticity
$(\nabla \mathbf{v})_p$	particles	matrix	grid-based velocity gradient
$\overline{\mathbf{v}}_p$	particles	vector	particle affine velocity field
$\mathbf{Z}_p$	particles	matrix → matrix	project to yield surface
m_i^n	grid	scalar	grid node mass
$\mathbf{v}_i^n$	grid	vector	rasterized velocity
$\overline{\mathbf{v}}_i^{n+1}$	grid	vector	final grid velocity, no friction
$\tilde{\mathbf{v}}_i^{n+1}$	grid	vector	final grid velocity
$\mathbf{v}_i^\star$	grid	vector	velocity with explicit forces
$\mathbf{x}_i^n$	grid	vector	Cartesian grid node locations
$\overline{\mathbf{x}}_i^{n+1}$	grid	vector	grid positions moved by $\overline{\mathbf{v}}_i^{n+1}$
$\mathbf{f}_i$	grid	matrix → vector	compute forces
λ_k	-	scalar	Lagrange multiplier for enforcing collison
G_k	-	vector → scalar	collision criterion
∇G_{ki}	grid	vector → vector	collision criterion gradient
$\hat{N}$	-	scalar → scalar	interpolation spline
N	-	vector → scalar	tensor product interpolation spline
∇N	-	vector → scalar	tensor product interpolation spline gradient
w_{ip}	mixed	scalar	interpolation weight
∇w_{ip}	mixed	vector	interpolation weight gradient

Table 4.1: †*These quantities are state on particles.*

Superscripts E and P are used to denote the elastic or plastic part of a deformation gradient ($\mathbf{F}_p^{E,n}$, $\mathbf{F}_p^{P,n}$).

Generally, quantities that live on particles, have indicators of time, and those that lack other adornments are state variables (α_p^n, $\mathbf{F}_p^{E,n}$, $\mathbf{x}_p^n$; not $\mathbf{v}_i^n$, $\hat{\mathbf{F}}_p^{E,n+1}$). There are two exceptions

to this. We do not store the deformation gradient itself, so $\mathbf{F}_p^n$ for us is not a state variable. The mass m_p is a state variable, but we omit a time indicator because it never changes. State variables are those that persist from the end of one time step to the beginning of the next.

Particle state. In MPM, the primary representation of state is stored on particles. We maintain mass m_p, position $\mathbf{x}_p^n$, velocity $\mathbf{v}_p^n$, and affine momentum $\mathbf{B}_p^n$, which is related to the velocity spatial derivatives. The extra matrix $\mathbf{B}_p^n$ stored per particle is used for APIC transfers [JSS15]. Up to a constant scale, this quantity approximates the spatial derivative of the grid velocity field at the end of the previous time step.

We also store the elastic and plastic components of the deformation gradient, $\mathbf{F}_p^{E,n}$ and $\mathbf{F}_p^{P,n}$. Note that while we use $\mathbf{F}_p^{E,n}$ to compute forces, $\mathbf{F}_p^{P,n}$ is not required and need not be stored. For plasticity, we must store one parameter α_p^n, which defines the size of the yield surface and may change per particle as a result of hardening.

Weights. We will frequently need to transfer information between particle and grid representations. We do this by associating with each particle p and grid node i a weight w_{ip}^n which determines how strongly the particle and node interact. If the particle and grid node are close together, the weight should be large. If the particle and node are farther apart, the weight should be small. We compute our weights based on a kernel as $w_{ip}^n = N(\mathbf{x}_p^n - \mathbf{x}_i^n)$, where $\mathbf{x}_p^n$ and $\mathbf{x}_i^n$ are the locations of the particle and grid node locations. We will also need the spatial derivatives of our weights, $\nabla w_{ip}^n = \nabla N(\mathbf{x}_p^n - \mathbf{x}_i^n)$, when we compute forces. We use time indices on the fixed grid node locations $\mathbf{x}_i^n$ to distinguish them from estimates (such as $\overline{\mathbf{x}}_i^{n+1}$) of where those nodes would end up if evolved with node velocities. We also indicate time on weights w_{ip}^n since they were computed using quantities at this time.

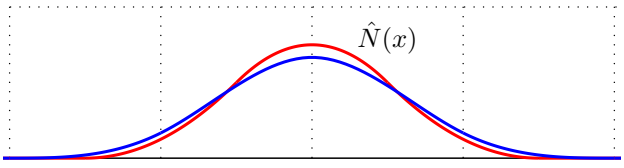


Figure 4.6: *Cubic (blue) and quadratic (red) splines used for computing interpolation weights.*

Choosing a kernel N leads to trade offs with respect to smoothness, computational efficiency, and the width of the stencil. We prefer tensor product splines for their computational efficiency, as they are relatively inexpensive to compute, differentiate, and store. The multilinear kernel typically employed for FLIP fluid solvers is the simplest of these options, but it is not suitable here. There are two reasons for this (see [SKB08]). The first is that ∇w_{ip}^n would be discontinuous and produce discontinuous forces. The second is that ∇w_{ip}^n may be far from zero when $w_{ip}^n \approx 0$, leading to large forces being applied to grid nodes with tiny weights. Quadratic and cubic b-splines work well, and we choose cubic b-splines for convenience. Our kernel is

$$\hat{N}(x) = \begin{cases} \frac{1}{2}|x|^3 - |x|^2 + \frac{2}{3} & 0 \le |x| < 1 \\ \frac{1}{6}(2 - |x|)^3 & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases} \tag{4.5}$$

$$N(\mathbf{u}) = \hat{N}\left(\frac{u_x}{h}\right)\hat{N}\left(\frac{u_y}{h}\right)\hat{N}\left(\frac{u_z}{h}\right), \tag{4.6}$$

where h is the grid spacing. Sometimes the quadratic kernel is also useful. We plot the quadratic and cubic kernels in Figure 4.6. We use the cubic spline for all of our examples.

Initialization. Particle locations are initialized with Poisson disk sampling. Initial values for m_p, $\mathbf{x}_p^n$, and $\mathbf{v}_p^n = \mathbf{v}(\mathbf{x}_p^n)$ are chosen based on the needs of the example, with $\mathbf{v}(\mathbf{x})$ the desired initial velocity field. $\mathbf{B}_p^n$ is initialized so that $\mathbf{C}_p^n = \mathbf{B}_p^n(\mathbf{D}_p^n)^{-1} = \nabla\mathbf{v}$ is the gradient of the initial velocity field and $\mathbf{D}_p^n$ is computed from (4.8). Our initial setups have no deformation, so $\mathbf{F}_p^{E,n} = \mathbf{F}_p^{E,n+1} = \mathbf{I}$. We initialize our hardening parameter with $q_p^n = 0$, from which we can compute α_p^n using (4.30) and (4.31). Initial particle volume V_p^0 is computed from the seeding density.

4.3.1 Transfer to grid

The first step of each time step is the transfer of state particles to the fixed Cartesian grid. We begin by distributing the mass of each particle to its neighboring grid nodes.

$$m_i^n = \sum_p w_{ip}^n m_p \tag{4.7}$$

Grid nodes far enough from any particle that they do not receive mass are inactive and do not participate in any further computations.

The next task is to transfer velocity. We do this using the APIC transfers in [JSS15]. The velocity state on the particle is represented by $\mathbf{v}_p^n$ and $\mathbf{B}_p^n$. The affine momentum $\mathbf{B}_p^n$ is related to the velocity spatial derivatives $\mathbf{C}_p^n$ through $\mathbf{C}_p^n = \mathbf{B}_p^n(\mathbf{D}_p^n)^{-1}$, where $\mathbf{D}_p^n$ is a matrix that behaves as an inertia tensor and is

$$\mathbf{D}_p^n = \sum_i w_{ip}^n(\mathbf{x}_i^n - \mathbf{x}_p^n)(\mathbf{x}_i^n - \mathbf{x}_p^n)^T = \begin{cases} \frac{h^2}{3}\mathbf{I} & \text{cubic} \\ \\ \frac{h^2}{4}\mathbf{I} & \text{quadratic} \end{cases} \tag{4.8}$$

where h is the grid spacing. Although the definition of the inertia tensor $\mathbf{D}_p^n$ depends on the relative positions of the grid nodes and particles through a relatively high-degree polynomial (both explicitly and through w_{ip}^n), it simplifies to a constant multiple of the identity in the cases of the quadratic and cubic splines presented.

With $\mathbf{C}_p^n$, we can define an affine velocity field $\overline{\mathbf{v}}_p(\mathbf{x})$ for particle p by $\overline{\mathbf{v}}_p(\mathbf{x}) = \mathbf{v}_p + \mathbf{C}_p^n(\mathbf{x} - \mathbf{x}_p^n)$. The momentum contribution from particle p to node i is $w_{ip}^n m_p \overline{\mathbf{v}}_p(\mathbf{x}_i^n)$. This leads to the full form of the velocity transfer,

$$\mathbf{v}_i^n = \frac{1}{m_i^n}\sum_p w_{ip}^n m_p(\mathbf{v}_p^n + \mathbf{B}_p^n(\mathbf{D}_p^n)^{-1}(\mathbf{x}_i^n - \mathbf{x}_p^n)) \tag{4.9}$$

4.3.2 Grid update

We next update velocities on the grid. This involves applying forces and processing for collisions with scripted objects. We present two approaches for doing this, explicit and

implicit. For most of our examples, explicit is more efficient, since we are running with relatively low stiffness. For stiff examples, implicit becomes advisable. We defer the implicit formulation until (§4.3.6).

Explicit. The simplest approach for handling forces is explicit. In this case, we compute and apply an explicit force (§4.4)

$$\mathbf{v}_i^\star = \mathbf{v}_i^n + \frac{\Delta t}{m_i^n}\mathbf{f}_i(\langle \mathbf{F}_p^{E,n} \rangle). \tag{4.10}$$

After forces are applied, we can process the velocities for collisions $\mathbf{v}_i^\star \to \overline{\mathbf{v}}_i^{n+1}$ and then apply friction $\overline{\mathbf{v}}_i^{n+1} \to \tilde{\mathbf{v}}_i^{n+1}$. The collision processing is described in (§4.6).

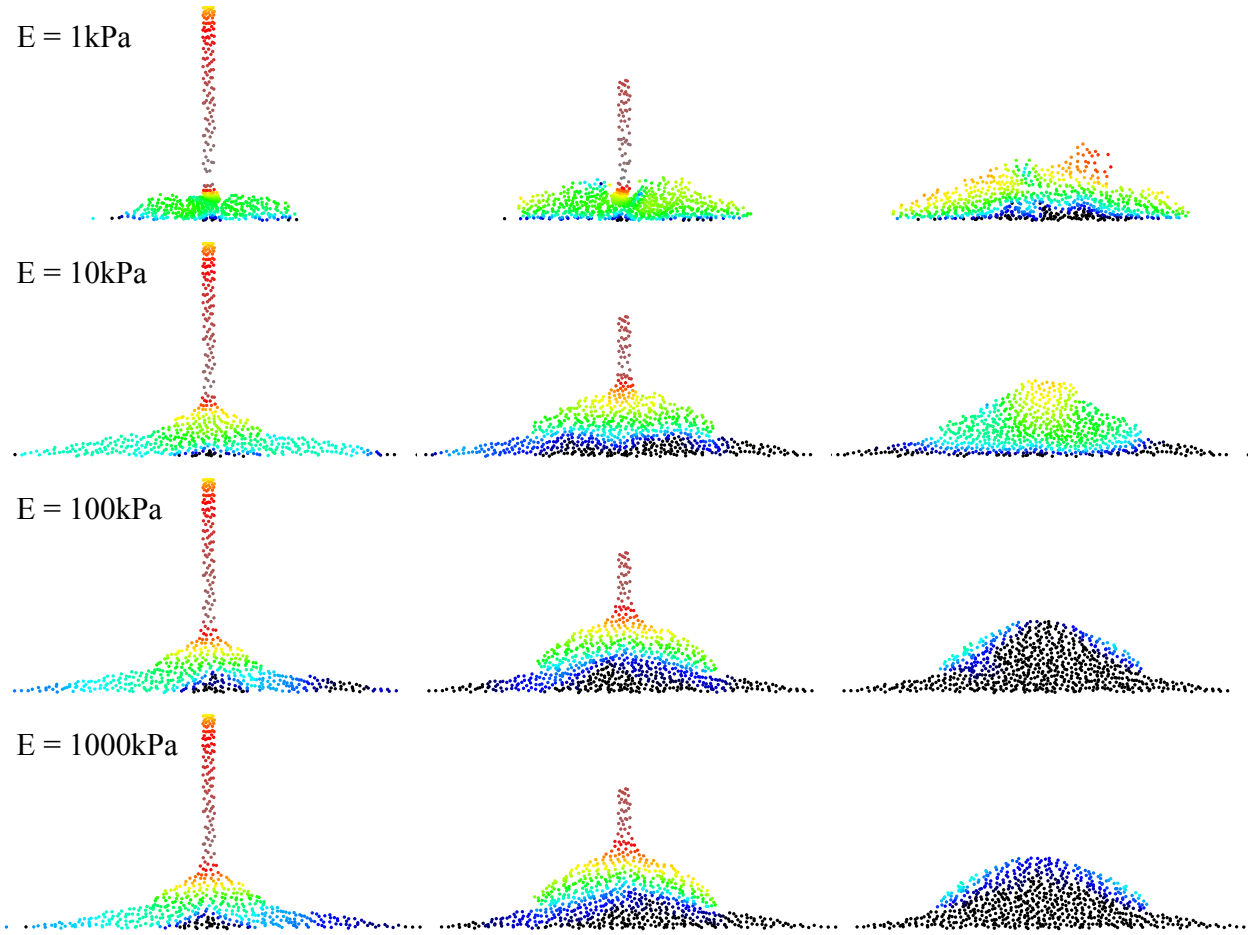


Figure 4.7: *Sand with a very low Young's modulus tends to be bouncy. The behavior is more like sand as the Young's modulus approaches its physical value.*

4.3.3 Transfer to particles

Next we transfer velocities from the grid back to particles. Since we are using APIC, we need to compute new velocities $\mathbf{v}_p^{n+1}$ and affine momentum $\mathbf{B}_p^{n+1}$. Velocities are interpolated back to particles in the straightforward way

$$\mathbf{v}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1}. \tag{4.11}$$

The transfer for $\mathbf{B}_p^{n+1}$ is

$$\mathbf{B}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} (\mathbf{x}_i^n - \mathbf{x}_p^n)^T. \tag{4.12}$$

4.3.4 Update particle state

Next, we update the particle's position and deformation gradient. Positions are updated by interpolating moving grid node positions

$$\mathbf{x}_p^{n+1} = \sum_i w_{ip}^n \overline{\mathbf{x}}_i^{n+1}. \tag{4.13}$$

Since the particles move with the flow, the material derivative in Equation 4.3 is just a normal time derivative and a simple difference yields the particle deformation gradient update

$$\mathbf{F}_p^{n+1} = \mathbf{F}_p^n + \Delta t (\nabla \mathbf{v})_p \mathbf{F}_p^n \tag{4.14}$$

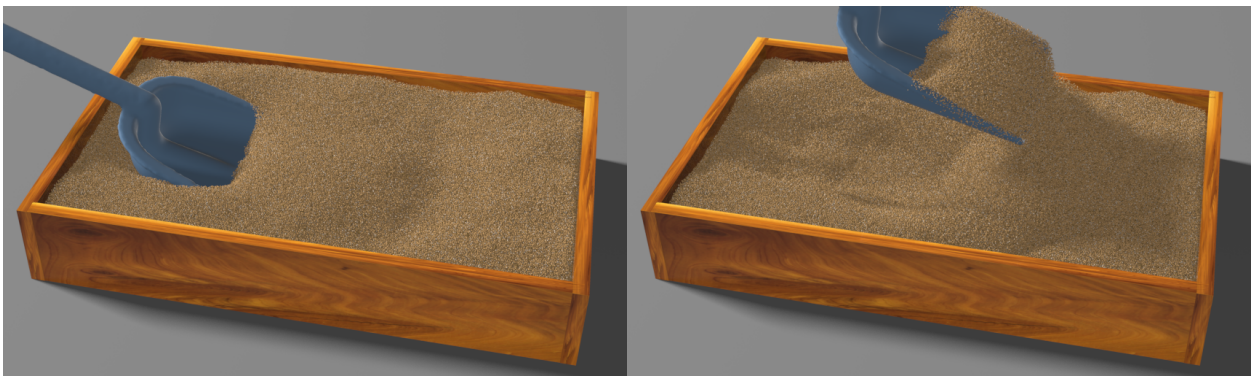


Figure 4.8: *A shovel digs through sand and pushes it aside.*

Figure 4.9: *A stick is dragged through a bed of sand, tracing out a butterfly shape in the sand.*

where $(\nabla \mathbf{v})_p$ is calculated by differentiating (4.11)

$$(\nabla \mathbf{v})_p = \sum_i \overline{\mathbf{v}}_i^{n+1} (\nabla w_{ip}^n)^T. \tag{4.15}$$

Note that we only store the elastic ($\mathbf{F}_p^{E,n}$) and plastic ($\mathbf{F}_p^{P,n}$) parts of $\mathbf{F}_p^n$ rather than $\mathbf{F}_p^n$ itself. These are related by $\mathbf{F}_p^n = \mathbf{F}_p^{E,n}\mathbf{F}_p^{P,n}$ (§4.5). During this evolution step, we assume that the plastic part is not changing ($\hat{\mathbf{F}}_p^{n+1} = \mathbf{F}_p^n$), which gives us the rule

$$\hat{\mathbf{F}}_p^{E,n+1} = \mathbf{F}_p^{E,n} + \Delta t (\nabla \mathbf{v})_p \mathbf{F}_p^{E,n}. \tag{4.16}$$

The plastic update is covered in (§4.5).

Note that the update of the particle position and deformation gradient use, $\overline{\mathbf{v}}_i^{n+1}$, while the velocity and related quantities use $\tilde{\mathbf{v}}_i^{n+1}$. The use of the frictional velocity in the updates of positional updates resulted in less stable behavior with implicit time stepping. With explicit time stepping, $\tilde{\mathbf{v}}_i^{n+1}$ could be used for both position and velocity related updates.

4.3.5 Plasticity, hardening

The final step is to apply plasticity and hardening. Plasticity is performed by projecting the elastic deformation gradient to its yield surface, an action denoted by $\mathbf{Z}(\cdot,\cdot)$ which we describe in detail later (§4.5). Plasticity does not change the full deformation gradient, so

that $\mathbf{F}_p^{n+1} = \hat{\mathbf{F}}_p^{E,n+1}\hat{\mathbf{F}}_p^{P,n+1} = \mathbf{F}_p^{E,n+1}\mathbf{F}_p^{P,n+1}$. This allows us to update the plastic part.

$$\mathbf{F}_p^{E,n+1} = \mathbf{Z}(\hat{\mathbf{F}}_p^{E,n+1}, \alpha_p^n) \tag{4.17}$$

$$\mathbf{F}_p^{P,n+1} = (\mathbf{F}_p^{E,n+1})^{-1}\hat{\mathbf{F}}_p^{E,n+1}\hat{\mathbf{F}}_p^{P,n+1} \tag{4.18}$$

The last step is hardening which updates $\alpha_p^n \to \alpha_p^{n+1}$ (§4.5.3).

4.3.6 Implicit velocity update

The implicit velocity update is

$$\overline{\mathbf{v}}_i^{n+1} = \mathbf{v}_i^n + \frac{\Delta t}{m_i^n}\mathbf{f}_i(\langle \mathbf{F}_p^{E,n+1} \rangle) + \sum_k \nabla G_{ki}\lambda_k \tag{4.19}$$

subject to the additional conditions $G_k \geq 0$, $\lambda_k \geq 0$, and $G_k\lambda_k = 0$. Here, $G_k(\langle \overline{\mathbf{x}}_i^{n+1} \rangle) \geq 0$, with $\overline{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n + \Delta t\overline{\mathbf{v}}_i^{n+1}$, is the collision-free criterion for all object-node collision pairs k. (§4.6) These forces are implicit, since $\mathbf{F}_p^{E,n+1}$ depends on $\overline{\mathbf{v}}_i^{n+1}$ through (4.16), (4.15), and (4.17). As in the explicit case, we complete the grid update by applying friction $\overline{\mathbf{v}}_i^{n+1} \to \tilde{\mathbf{v}}_i^{n+1}$ and described in (§4.6).

Note that we are implicit in plasticity, but we are not implicit in hardening or friction. In the absence of plasticity, these are just the Karush-Kuhn-Tucker (KKT) conditions [NW06] for minimization. Unlike solving a minimization problem, however, our linear systems are not generally symmetric, and we do not have an objective with which to do line searches.

Solving the system. Since the collision constraints are independent, we use the projection method to eliminate the collisions. We solve the nonlinear system of equations using Newton's method. Note that because of plasticity, the systems will in general be asymmetric, and we solve with GMRES. These systems usually converge sufficiently in three or fewer iterations of GMRES, rarely ($< 1\%$) taking more than four iterations. We limit GMRES to 15 iterations and allow multiple Newton iterations.

4.3.7 Initialization

Particle locations are initialized with Poisson disk sampling. Initial values for m_p, $\mathbf{x}_p^n$, and $\mathbf{v}_p^n = \mathbf{v}(\mathbf{x}_p^n)$ are chosen based on the needs of the example, with $\mathbf{v}(\mathbf{x})$ the desired initial velocity field. $\mathbf{B}_p^n$ is initialized so that $\mathbf{C}_p^n = \mathbf{B}_p^n(\mathbf{D}_p^n)^{-1} = \nabla\mathbf{v}$ is the gradient of the initial velocity field and $\mathbf{D}_p^n$ is computed from (4.8). Our initial setups have no deformation, so $\mathbf{F}_p^{E,n} = \mathbf{F}_p^{E,n+1} = \mathbf{I}$. We initialize our hardening parameter with $q_p^n = 0$, from which we can compute α_p^n using (4.30) and (4.31). Initial particle volume V_p^0 is computed from the seeding density.

4.4 Forces

Here we derive the MPM forces on Eulerian grid nodes $\mathbf{f}_i(\langle\mathbf{F}_p^E\rangle)$. These forces are obtained by differentiating a discretization of the potential energy with respect to the motion of grid nodes.

Figure 4.10: *Sand is poured from a spout into a pile in a lab (left) and with our method (right).*

4.4.1 Continuous setting

Elastic materials are characterized by their ability to store potential energy and then release it by doing work to cause motion (kinetic energy). Let Ψ be the total potential energy stored by a material at a given time. In a real material, potential energy is stored locally in response to deformation. This is called *energy density*, or energy per unit volume, and represented by ψ. Since this depends only on the local deformation, we can write $\psi(\mathbf{F})$. The function $\psi(\mathbf{F})$ captures the essential information about the way an elastic material responds to deformation. This relationship depends on the material; we choose our model in (§4.4.3).

In much the same way that total mass is computed by integrating the density of a material over its volume Ω, potential energy is computed by integrating energy density $\Psi = \int_\Omega \psi \, dV$.

4.4.2 Discrete setting

We discretize the potential energy with a sum on particles,

$$\Psi = \sum_p V_p^0 \psi(\mathbf{F}_p^E) \tag{4.20}$$

Note that V_p^0 is the volume of material attributed to a particle in the initial configuration. Only the elastic portion of the deformation gradient $\mathbf{F}_p^E$ contributes to the energy [BW08].

If the state of the system is described by a finite number of positions $\mathbf{x}_1, \ldots, \mathbf{x}_m$ (picture a bunch of point masses connected by springs), then the potential energy can be written $\Phi(\mathbf{x}_1, \ldots, \mathbf{x}_m)$. Moving one of these points causes the amount of energy to change (energy is required to stretch or compress the springs). The springs will push back on these points so as to release this built-up energy. In this way, the force felt by particle j will be $\mathbf{f}_j = -\frac{\partial \Psi}{\partial \mathbf{x}_j}$.

With MPM, grid nodes are temporarily Lagrangian, and can be moved to define the force. If the current grid node velocity is $\mathbf{v}_i$, then its position can be approximated as $\mathbf{x}_i = \mathbf{x}_i^n + \Delta t \mathbf{v}_i$. Considering a different ending position implies a different velocity to get there. These node velocities are used in (4.15) to compute $(\nabla \mathbf{v})_p$, which is in turn used by (4.16) to compute a new deformation gradient $\mathbf{F}_p^E$. This deformation gradient will be used to compute energy

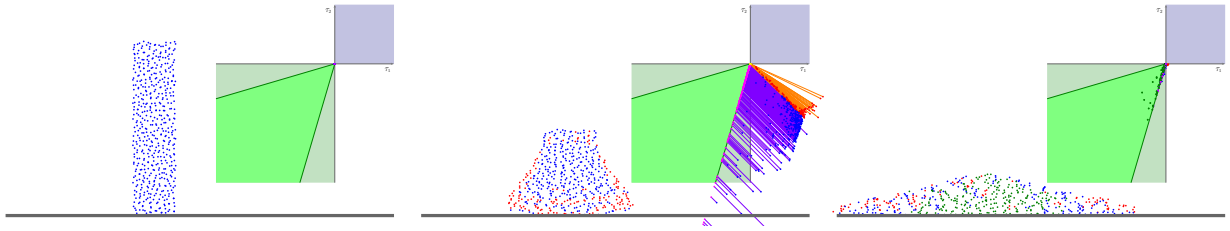


Figure 4.11: *Sand particles are colored based on their current plastic deformation behavior. The plot shows the locations of these particles in principal stress space. Green particles lie within the yield surface and experience no plasticity. Blue particles are projected to the yield surface along a direction that avoids volume change. Red particles are experiencing tension and are projected to the tip of the conical yield surface; these particles are separating freely with no stress.*

density using a model $\psi(\mathbf{F}_p^E)$, which finally gives us total potential energy. In this way, the potential energy of the material can be expressed in terms of the locations of the grid nodes. We can use this relationship, summarized below, to compute forces on grid nodes.

$$\Psi(\langle \mathbf{x}_i \rangle) = \sum_p V_p^0 \psi(\mathbf{F}_p^E(\langle \mathbf{x}_i \rangle)) \tag{4.21}$$

$$\mathbf{F}_p^E(\langle \mathbf{x}_i \rangle) = \left(\mathbf{I} + \sum_i (\mathbf{x}_i - \mathbf{x}_i^n)(\nabla w_{ip}^n)^T\right)\mathbf{F}_p^{E,n} \tag{4.22}$$

This relationship can be differentiated to deduce the desired equation for computing grid node forces

$$\mathbf{f}_i(\langle \mathbf{F}_p^E \rangle) = -\frac{\partial \Psi}{\partial \mathbf{x}_i} = -\sum_p V_p^0 \left(\frac{\partial \psi}{\partial \mathbf{F}}(\mathbf{F}_p^E)\right)(\mathbf{F}_p^{E,n})^T \nabla w_{ip}^n. \tag{4.23}$$

Note that $\mathbf{F}_p^E$ is the function parameter but $\mathbf{F}_p^{E,n}$ is a known value which is not changing during the current time step. Also note that all deformation is assumed to be elastic. When computing the force, the effect of further plastic flow is ignored [BW08].

Gravity. We include gravity as an additional term in (4.23)

$$\mathbf{f}_i^{grav} = \sum_p w_{ip}^n m_p \mathbf{g} = m_i^n \mathbf{g}. \tag{4.24}$$

4.4.3 Constitutive model

We adopt the energy density $\psi(\mathbf{F})$ from Mast et al. [Mas13]. This model uses the same energy density as St. Venant-Kirchhoff, but it replaces the left Cauchy Green strain with the Hencky strain $\frac{1}{2}\log(\mathbf{F}\mathbf{F}^T)$. This not only removes the failures of traditional St. Venant Kirchhoff under large deformation, but also makes a number of aspects of the Drucker-Prager plastic projection very simple (see (§4.14.5)). The model is most conveniently written in terms of the singular value decomposition $\mathbf{F} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$ as

$$\psi(\mathbf{F}) = \mu \operatorname{tr}\left((\log \boldsymbol{\Sigma})^2\right) + \frac{1}{2}\lambda(\operatorname{tr}(\log \boldsymbol{\Sigma}))^2, \tag{4.25}$$

where $\boldsymbol{\Sigma}$ is diagonal so $\log \boldsymbol{\Sigma}$ is computed by taking the logarithm of the diagonal entries. The force computation (4.23) requires the derivative of this, which is

$$\frac{\partial \psi}{\partial \mathbf{F}}(\mathbf{F}) = \mathbf{U}(2\mu\boldsymbol{\Sigma}^{-1}\log \boldsymbol{\Sigma} + \lambda \operatorname{tr}(\log \boldsymbol{\Sigma})\boldsymbol{\Sigma}^{-1})\mathbf{V}^T. \tag{4.26}$$

4.5 Plasticity

Elastic and plastic deformation gradient. We represent plasticity by factoring deformation gradient into elastic and plastic parts as $\mathbf{F}_p^n = \mathbf{F}_p^{E,n}\mathbf{F}_p^{P,n}$. The deformation gradient is a measure of how a material has locally rotated and deformed due to its motion. By factoring the deformation gradient in this way, we divide this deformation history into two pieces. The plastic part, $\mathbf{F}_p^{P,n}$, represents the portion of the material's history that has been forgotten. If a metal rod is bent into a coiled spring, the rod *forgets* that it used to be straight; the coiled spring behaves as though it was always coiled (see Figure 4.3). The twisting and bending involved in this operation is stored in $\mathbf{F}_p^{P,n}$. If the spring is compressed slightly, the spring will feel strain (deformation). This is elastic deformation, which is stored in $\mathbf{F}_p^{E,n}$. The spring remembers this deformation. In response, the material exerts stress to try to restore itself to its coiled shape. In this way, we see that only $\mathbf{F}_p^{E,n}$ should be used to compute stress. The full history of the metal rod consists of being bent into a spring shape

($\mathbf{F}_p^{P,n}$) and then being compressed ($\mathbf{F}_p^{E,n}$).

4.5.1 Projecting to the yield surface

The only algorithmic aspect of our plasticity treatment that has not yet been defined is our function $\mathbf{Z}(\mathbf{F}_p^E, \alpha_p)$, which projects the deformation gradient $\mathbf{F}_p^E$ to the yield surface defined by the parameter α_p. The Drucker-Prager plasticity model is based on Coulomb friction interactions between sand particles. In the continuum setting, this means that the shear stress cannot be larger than a coefficient of friction times the normal stress. This results in a simple constraint on the principal stresses which we derive in (§4.11.1).

In the space of principal stress, the yield surface looks like a cone (see Figure 4.13). There are three possible cases that must be considered. If the stress lies within the yield

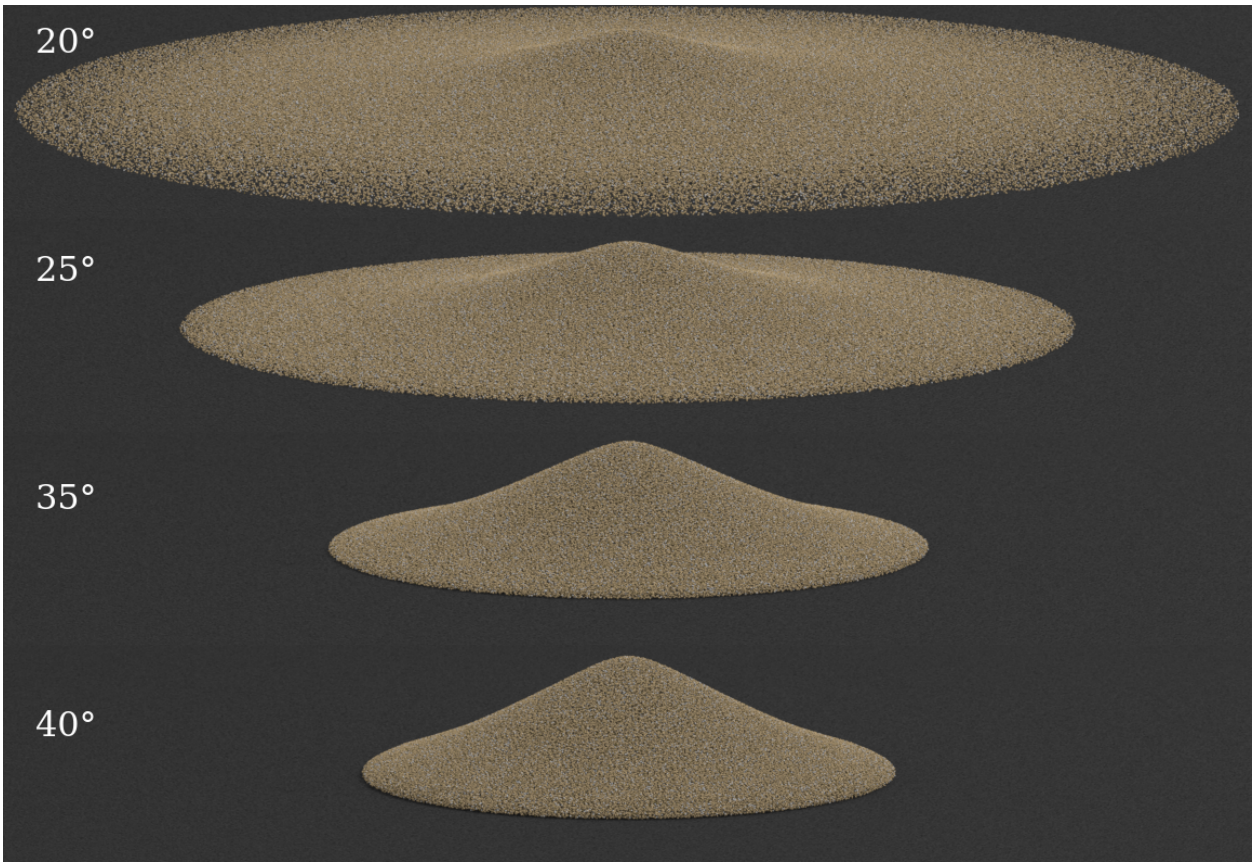


Figure 4.12: *Varying the friction angle changes the shape of a pile of sand. A larger angle produces a taller sand pile with steeper sides.*

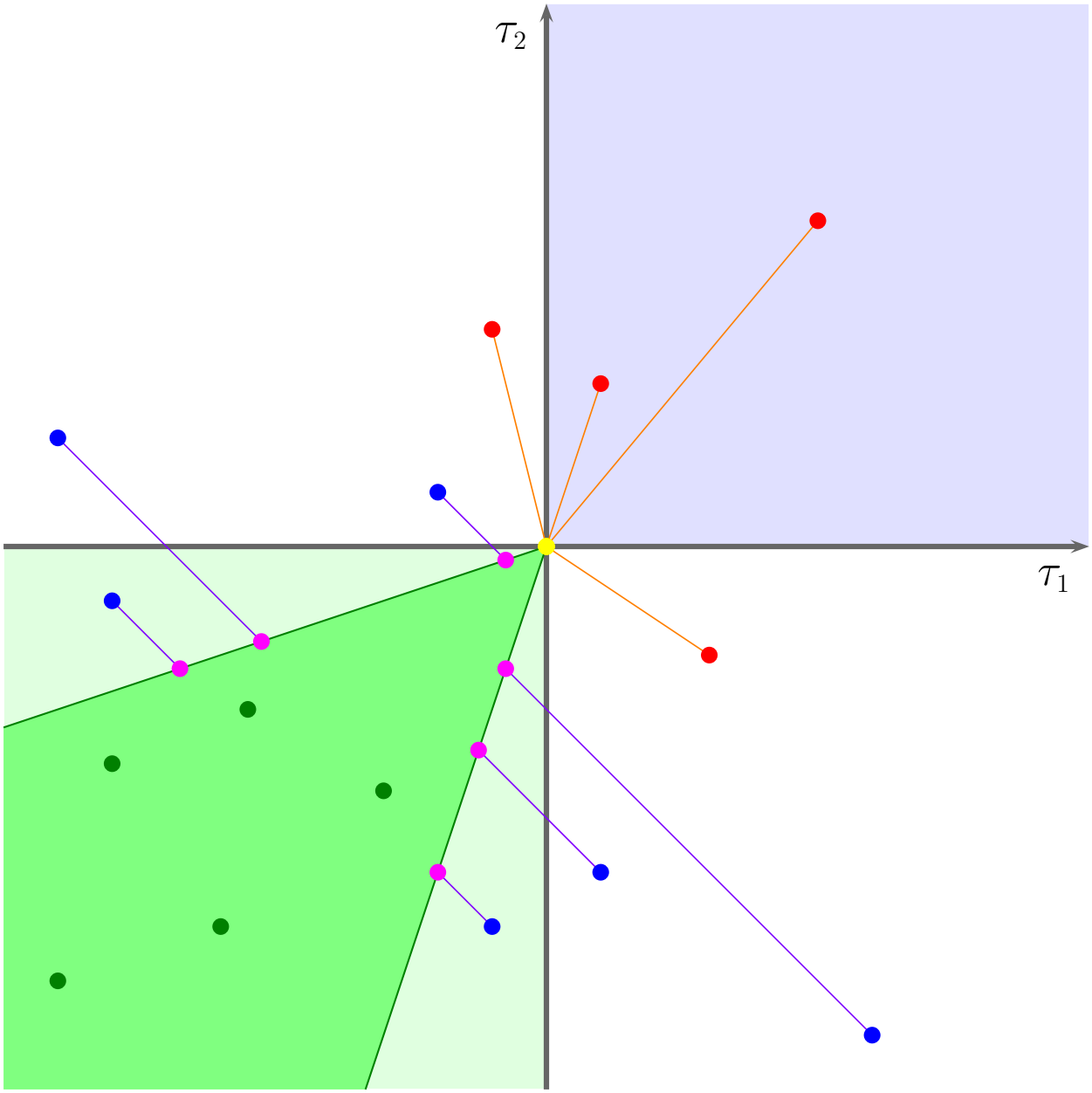


Figure 4.13: *The yield surface has the shape of a cone with its tip at the origin, which corresponds to no stress. Green particles are inside the yield surface and exhibit an elastic response. Blue particles are under compression but experience more shear than friction allows. These configurations are projected to the yield surface along a direction that avoids volume change. Red particles are experiencing tension and are projected to the tip of the conical yield surface. These particles separate freely without stress.*

surface (Case I), then there is static friction between sand particles, and no plasticity occurs. If the sand is undergoing expansion (Case II), then there is no resistance to motion; this corresponds to the tip of the cone. Otherwise, there is dynamic friction (Case III), and we

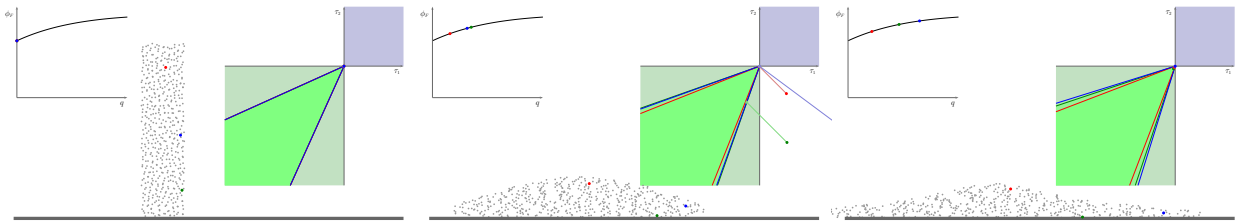


Figure 4.14: *Three particles in a collapsing pile of sand are colored for reference. As these particles deform plasticly, their yield surface changes as they undergo hardening, resulting in a wider cone for projection. Hardening causes each particle to have its own yield surface.*

should project to the side of the cone. Examples of these cases in an actual simulation can be seen in Figure 4.11.

As with energy density, plasticity is most conveniently defined in terms of the singular value decomposition of the deformation gradient, $\mathbf{F}_p^E = \mathbf{U}_p \mathbf{\Sigma}_p \mathbf{V}_p{}^T$. Let $\boldsymbol{\varepsilon}_p = \log \mathbf{\Sigma}_p$ and

$$\hat{\boldsymbol{\varepsilon}}_p = \boldsymbol{\varepsilon}_p - \frac{\operatorname{tr}(\boldsymbol{\varepsilon}_p)}{d}\mathbf{I} \qquad \delta\gamma_p = \|\hat{\boldsymbol{\varepsilon}}_p\|_F + \frac{d\lambda + 2\mu}{2\mu}\operatorname{tr}(\boldsymbol{\varepsilon}_p)\alpha_p \tag{4.27}$$

where d is the spatial dimension and $\delta\gamma_p$ is the amount of plastic deformation. If $\delta\gamma_p \le 0$, then the candidate $\mathbf{F}_p^E$ is already in the yield surface and should be returned without modification (Case I). If $\|\hat{\boldsymbol{\varepsilon}}_p\|_F = 0$ or $\operatorname{tr}(\boldsymbol{\varepsilon}_p) > 0$, then we need to project to the cone's tip (Case II), in which case we should return $\mathbf{U}_p \mathbf{V}_p{}^T$. Otherwise, we should project to the cone surface (Case III) by returning $\mathbf{U}_p e^{\mathbf{H}_p} \mathbf{V}_p{}^T$, where

$$\mathbf{H}_p = \boldsymbol{\varepsilon}_p - \delta\gamma_p \frac{\hat{\boldsymbol{\varepsilon}}_p}{\|\hat{\boldsymbol{\varepsilon}}_p\|_F} \tag{4.28}$$

Note that the operations $\log \mathbf{\Sigma}_p$ and $e^{\mathbf{H}_p}$ involve diagonal matrices, so that the logarithm and exponential functions are simply applied to the diagonal elements. Note also that the result of this projection $\mathbf{Z}$ has a straightforward singular value decomposition ($\mathbf{U}_p$ and $\mathbf{V}_p$ do not change), and this decomposition will be required when computing the force. We avoid the extra decomposition by returning the diagonal part ($\mathbf{\Sigma}_p$, $\mathbf{I}$, or $e^{\mathbf{H}_p}$) rather than the full result ($\mathbf{F}_p^E$, $\mathbf{U}_p \mathbf{V}_p{}^T$, or $\mathbf{U}_p e^{\mathbf{H}_p} \mathbf{V}_p{}^T$).

4.5.2 Note on preventing undesired volume change

The method as described has the desirable feature that sand is prevented from compressing arbitrarily as a byproduct of losing volume in the plasticity projection. To see this, note that a change in $\det(\mathbf{F}_p^E)$ corresponds to a change in volume of the elastic deformation. In Case I, $\mathbf{F}_p^E$ is unchanged, so volume is not changed. In Case II, the sand expands, and volume should be gained. In Case III, the sand deforms plasticly and an associative flow rule [BW08] would lead to excessive volume gain. Instead, noting that $\mathrm{tr}(\hat{\boldsymbol{\varepsilon}}_p) = 0$, the Drucker-Prager model uses a non-associative flow to preserve volume during the plastic projection

$$\det(\mathbf{U}_p e^{\mathbf{H}_p} \mathbf{V}_p{}^T) = e^{\mathrm{tr}(\mathbf{H}_p)} = e^{\mathrm{tr}(\boldsymbol{\varepsilon}_p)} = \det(\boldsymbol{\Sigma}_p) = \det(\mathbf{F}_p^E).$$

The key to retaining volume in this case is to ensure that $\mathrm{tr}(\mathbf{H}_p) = \mathrm{tr}(\boldsymbol{\varepsilon}_p)$, which means the projection to the cone should locate the closest point on the cone that does not change the trace, rather than the closest point on the cone. We discuss this in more detail in (§4.12.2).

4.5.3 Hardening

We adopt the hardening model of Mast et al. [MAM14], where plastic deformation can increase the friction between sand particles. The amount of hardening depends on the amount of correction that occurred due to plasticity. In Case I, no plasticity occurred, so $\delta q_p = 0$. In Case II, all of the stress was removed, so $\delta q_p = \|\boldsymbol{\varepsilon}_p^{E,n+1}\|_F$. In Case III, the amount of plasticity that occurred was $\delta q_p = \delta\gamma_p$. In each case, $\delta q_p \geq 0$. We define our hardening update using

$$q_p^{n+1} = q_p^n + \delta q_p \tag{4.29}$$

$$\phi_{Fp} = h_0 + (h_1 q_p^{n+1} - h_3) e^{-h_2 q_p^{n+1}} \tag{4.30}$$

$$\alpha_p^{n+1} = \sqrt{\frac{2}{3}} \frac{2\sin\phi_{Fp}}{3 - \sin\phi_{Fp}} \tag{4.31}$$

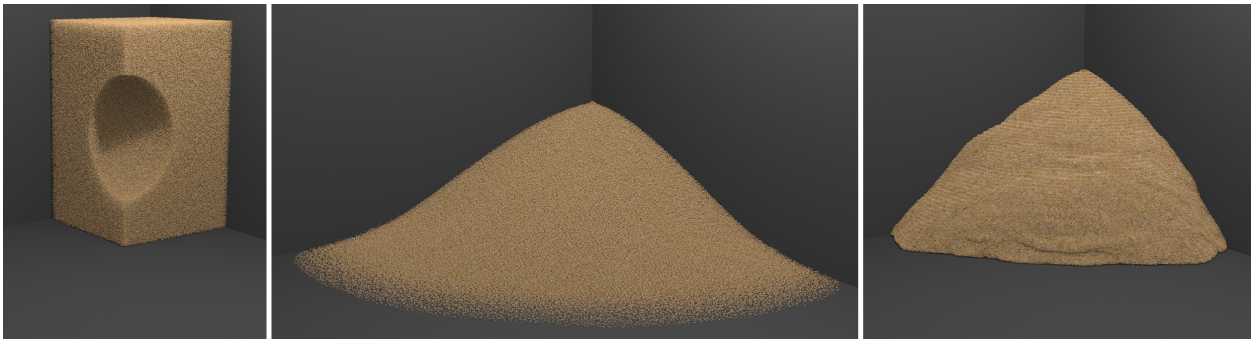


Figure 4.15: *Comparison on notched sand block fall. Initial (left), ours (middle), and Narain et al. [2010] (right).*

The quantity q_p^n is the hardening state, ϕ_{F_p} is often referred to as the friction angle, the *internal coefficient of friction* is $\tan\phi_{F_p}$, and (4.30) models a curve with a maximum and an asymptote. Plausible values of ϕ_{F_p} lie in $[0, \frac{\pi}{2})$, with $\phi_{F_p} = 0$ behaving as a fluid. Feasible hardening parameters satisfy $h_0 > h_3 \geq 0$ and $h_1, h_2 \geq 0$. The values we use are listed in Table 4.2. Figure 4.14 illustrates the change in yield surface as particles undergo hardening in 2D.

4.6 Collisions

We separate our collision response into two distinct steps: resolving the actual collision and applying friction. The motivation for this is that the collision response can be added into the implicit solve, but doing the same for friction would be more difficult. In the explicit case, this separation does not matter.

We use a signed distance function $\phi(\mathbf{x})$ to represent each obstacle, with the convention that negative is inside the object and positive is outside. If we could process particles for collisions directly, then the collision constraint would be $\phi(\mathbf{x}_p^{n+1}) \geq 0$. In practice, processing collisions directly on particles produces poor results, since it causes $\mathbf{x}_p^{n+1}$ and $\mathbf{F}_p^{E,n+1}$ to get out of sync. This can cause objects to slowly seep into the ground. Instead, it is necessary to process collisions using the grid velocities.

Since $\mathbf{x}_p^{n+1}$ will be computed based on $\overline{\mathbf{v}}_i^{n+1}$, one could adjust $\overline{\mathbf{v}}_i^{n+1}$ to enforce $\phi(\mathbf{x}_p^{n+1}) \geq 0$. While this would likely lead to good results, it complicates collision processing in both the

explicit and implicit cases. Instead, we process collisions against the nodes themselves as in [GSS15]. This is difficult because $\phi(\overline{\mathbf{x}}_i^{n+1}) \geq 0$ does not make sense. There *should* be grid nodes inside obstacles. A set of constraints $G_k(\langle\overline{\mathbf{x}}_i^{n+1}\rangle) \geq 0$ or $G_k(\langle\overline{\mathbf{x}}_i^{n+1}\rangle) = 0$ on grid nodes is needed that avoid collisions for particles, at least approximately, but which can be applied independently per grid node in an straightforward manner. This depends on the type of collision being applied. We support three types of collisions: sticky, slipping, and separating. Note that a grid node must have received mass during the transfer in order to be considered for a collision constraint of any type.

Sticky. Sticky collisions enforce that a point remains fixed to a particular reference point on the collision object. We enforce this by requiring $\overline{\mathbf{v}}_i^{n+1} = \mathbf{v}_b^{n+1}$, where $\mathbf{v}_b^{n+1}$ is the velocity of the collision object at the candidate position. In terms of positions, this is $G_k(\overline{\mathbf{x}}_i^{n+1}) = \overline{\mathbf{x}}_i^{n+1} - \mathbf{x}_i^n - \Delta t \mathbf{v}_b^{n+1} = 0$. The constraint can be enforced by directly setting the velocity.

Separating. Separating constraints have two cases. If a node is already inside a collision body ($\phi(\mathbf{x}_i^n) < 0$), then it should not penetrate any deeper, $\phi(\overline{\mathbf{x}}_i^{n+1}) \geq \phi(\mathbf{x}_i^n)$. If a node is originally outside the object ($\phi(\mathbf{x}_i^n) \geq 0$) then it should remain $\phi(\overline{\mathbf{x}}_i^{n+1}) \geq 0$. These cases can be combined into the constraint $\phi(\overline{\mathbf{x}}_i^{n+1}) \geq \min(\phi(\mathbf{x}_i^n), 0)$. Note that movement along and away from the collision object are fully permitted by this rule, even if the collision surface is curved. An unsatisfied constraint of the form $\phi(\mathbf{x}_i) \geq a$ or $\phi(\mathbf{x}_i) = a$ can be enforced by $\mathbf{x}_i \leftarrow \mathbf{x}_i - (\phi(\mathbf{x}_i) - a)\nabla\phi(\mathbf{x}_i)$, noting that $\nabla\phi$ is the normal direction.

Slipping. For a slipping constraint, we do not want to allow separation for existing collisions, but sliding along the surface is permitted. If a node is already inside a collision body ($\phi(\mathbf{x}_i^n) < 0$), then it should stay at its current depth, $\phi(\overline{\mathbf{x}}_i^{n+1}) = \phi(\mathbf{x}_i^n)$. If a node is originally outside the object ($\phi(\mathbf{x}_i^n) \geq 0$) then no collision constraint is enforced. By not enforcing this constraint for non-penetrating nodes, penetration becomes possible and leads to enforcement in the next time step. Slipping constraints are enforced as in the separating case.

With a mathematical description for the constraints for all cases and a method for directly

enforcing those constraints, direct enforcement ($\mathbf{v}_i^\star \to \overline{\mathbf{v}}_i^{n+1}$) is all that is required for the explicit case. The implicit case uses the constraints G_k that have been defined in order to couple collision enforcement with force application (§4.3.6).

4.6.1 Friction

To apply friction, we look not at the manner in which collisions were enforced but the effect that this enforcement had on the velocities. In the explicit case, velocities before ($\mathbf{v}_i^\star$) and after ($\overline{\mathbf{v}}_i^{n+1}$) are already available. $\Delta\mathbf{v}_i = \overline{\mathbf{v}}_i^{n+1} - \mathbf{v}_i^\star$ is the velocity change attributable to collisions.

In the implicit case, the collision contribution is from the last term of Equation 4.19. We compute the velocity estimate before forces as $\mathbf{v}_i^\star = \mathbf{v}_i^n + \frac{\Delta t}{m_i^n}\mathbf{f}_i(\langle \mathbf{F}_p^{E,n+1} \rangle)$. Although $\overline{\mathbf{v}}_i^{n+1}$ would be the after-collision velocity if the implicit solve had converged, this is not often done in practice. Instead, we repeat collision processing on $\mathbf{v}_i^\star$ to compute the difference for $\Delta\mathbf{v}_i$.

For both cases, $\Delta\mathbf{v}_i$ is the velocity change that collisions caused. Corresponding to this, an impulse $\mathbf{j} = m_i^n \Delta\mathbf{v}_i$ must have been applied. Since each node participates in at most one collision (the constraints do not mix), the normal direction $\mathbf{n}$ is known. (If it were not, it could be approximated as $\mathbf{n} = \frac{\mathbf{j}}{\|\mathbf{j}\|}$.) This divides velocity into normal and tangential parts: $v_{in} = \mathbf{n} \cdot \overline{\mathbf{v}}_i^{n+1}$ and $\mathbf{v}_{it} = \overline{\mathbf{v}}_i^{n+1} - \mathbf{n}v_{in}$. The tangential direction is $\mathbf{t} = \frac{\mathbf{v}_{it}}{\|\mathbf{v}_{it}\|}$. The Coulomb friction law limits the amount of friction that can be applied to $\mu_b\|\mathbf{j}\|$, where μ_b is the coefficient of friction. If $\|\mathbf{v}_{it}\| \le \frac{\mu_b}{m_i^n}\|\mathbf{j}\|$, then friction suffices to eliminate tangential motion entirely, and $\tilde{\mathbf{v}}_i^{n+1} = \mathbf{n}v_{in}$. Otherwise, $\tilde{\mathbf{v}}_i^{n+1} = \overline{\mathbf{v}}_i^{n+1} - \frac{\mu_b}{m_i^n}\|\mathbf{j}\|\mathbf{t}$.

4.7 Rendering

Zhu and Bridson [ZB05] render sand with a reconstructed surface. Narain et al. [NGL10] associate a number of render points sampled near each particle for high resolution rendering. In our examples we have a sufficient number of simulated particles to simply render each particle as a matte sphere. The color of each particle is randomly chosen from yellow (RGB

	ρ	E	ν	Friction Angle	Hardening Parameters
Castle	2200	3.537×10^5	0.3	—	35/0/0.2/10
Effect of friction angle	2200	3.537×10^5	0.3	20/25/30/35/40	—
Hourglass	2200	3.537×10^5	0.3	—	35/9/0.3/10
Butterfly	2200	3.537×10^5	0.3	—	35/9/0.2/10
Butterfly close	2200	3.537×10^5	0.3	—	35/9/0.2/10
Raking	2200	3.537×10^5	0.3	—	35/9/0.2/10
Raking close	2200	3.537×10^5	0.3	—	35/9/0.2/10
Pile from spout	2200	3.537×10^5	0.3	30	—
Splash	1582	3.537×10^6	0.3	22	—
Shovel	2200	3.537×10^5	0.3	—	35/9/0.2/10
Effect of Young's Modulus	2200	$10^{3,4,5,6}$	0.3	—	35/9/0.3/10

Table 4.2: *Sand material parameters. Friction angle ϕ_F and hardening parameters h_0, h_1, and h_3 are listed in degrees for convenience; all formulas in the text use radians.*

225/169/95 with a probability of 0.85), brown (RGB 107/84/30 with a probability of 0.1) and white (RGB 255/255/255 with a probability of 0.05) to further improve realism. All scenes were rendered using SideFX's Mantra. For scenes with rapidly flowing sand (such as the hourglass) we turned on motion blur where appropriate.

4.8 Results

Flowing and Piling. We demonstrate the accuracy of our model by showing the characteristic behaviors of sand flowing and piling. In Figure 4.1, we simulate sand flowing inside an hourglass. The sand forms a smooth granular flow and piles up at the bottom. Figure 4.10 shows a stream of sand inflow hitting a high frictional surface. We compare this simulation with real world footage. Our model successfully captures the interesting avalanche instability [Yos03] of this experiment.

Easy Tuning. We list the parameters used in our examples in Table 4.2 and the runtime performance of those simulations in Table 4.3. In Figure 4.12, we simulate columns of dry sand with different friction angles collapsing on the ground. Different friction angles directly affect the interaction between sand grains, therefore the final piling angle. While the real Young's modulus of sand is 3.537×10^7, we found that sometimes choosing a moderately smaller value does not change the visual appearance. In Figure 4.7, we show 2D inflow simulations with different Young's modulus. A moderately smaller Young's modulus improves the efficiency of the implicit solve. However, the material may exhibit jiggling behavior if it

is too small. We assert physically accurate Young's modulus is always the best choice unless an artistic elastic effect is desirable.

Two-way Coupling. The benefits of using MPM include automatic self collision and coupling between different materials. In Figure 1.7, we show an elastic ball interacting with a dry sand castle. MPM naturally handles the two-way coupling without requiring any additional treatment other than assigning different constitutive models to different particles.

Drawing and Scooping. We further demonstrate the versatility of our method by performing various tasks in a sand box. Figure 4.9 shows drawing a butterfly with a wooden stick. Figure 4.4 shows raking sand in a Zen garden. Figure 4.8 shows scooping sand.

APIC Stability. Our method benefits from the APIC particle/grid transfers [Jia15, JSS15] due to its stability and low numerical dissipation. In Figure 4.16, we run a 2D inflow simulation and compare our result with Mast et al. [Mas13, MAM14] where a traditional FLIP transfer scheme is used. With the same material parameters and time step sizes, our method does not suffer from the unstable ringing instability like FLIP does. We further show the robustness and stability of our method in a 3D energetic scenario. In Figure 1.5, a rigid ball is dropped into a $1m \times 1m \times 0.35m$ sand box with impact speed $6m/s$. The impact dynamics are stable and almost noise-free, resulting in a smooth and symmetric crown splash visual appearance.

	Scheme	FPS	Min/Frame	Δt	Particles	Threads	CPU	Δx	Grid
Castle	Implicit	72	6.80	1×10^{-3}	7.95×10^5	4	†2.67GHz	0.01	$300 \times 140 \times 200$
Friction angle	Explicit	120	4.10	1×10^{-4}	1.20×10^6	4	‡3.33GHz	0.001	$432 \times 144 \times 432$
Hourglass	Explicit	48	2.00	1×10^{-4}	4.60×10^5	12	⋆3.00GHz	0.0025	$160 \times 360 \times 160$
Butterfly	Explicit	24	10.31	2.5×10^{-4}	3.84×10^6	10	⋆3.00GHz	0.0045	$220 \times 55 \times 220$
Butterfly close	Explicit	48	21.23	1×10^{-4}	4.10×10^6	8	⋆3.00GHz	0.0014	$280 \times 140 \times 210$
Raking	Explicit	24	9.78	2.5×10^{-4}	3.84×10^6	10	⋆3.00GHz	0.0045	$220 \times 55 \times 220$
Raking close	Explicit	24	32.84	1×10^{-4}	4.30×10^6	8	°2.90GHz	0.0021	$336 \times 96 \times 288$
Pile from spout	Implicit	120	4.34	1.5×10^{-4}	9.94×10^5	8	†2.67GHz	0.000 83	$240 \times 96 \times 240$
Splash	Explicit	240	9.27	5×10^{-5}	6.60×10^6	8	§3.47GHz	0.0078	$256 \times 256 \times 256$
Shovel	Explicit	24	24.8	1×10^{-4}	1.96×10^6	4	§3.47GHz	0.005	$160 \times 100 \times 100$
E c	Implicit	24	4.9×10^{-3}	7.5×10^{-4}	7.40×10^2	1	†2.67GHz	0.016	256×64

Table 4.3: *Sand simulation performance. Note that Δt denotes the maximum allowed time step size. The actual Δt is adaptive and may be restricted by CFL condition when the particle velocities are high. In all of our simulations we use a CFL number of 1, i.e., we don't allow particles to move further than Δx in a time step. CPU types used are: †Intel Xeon X5650, ‡Intel Xeon W3680, ⋆Intel Xeon E5-2690 v2, °Intel Xeon E5-2690, §Intel Xeon X5690.*

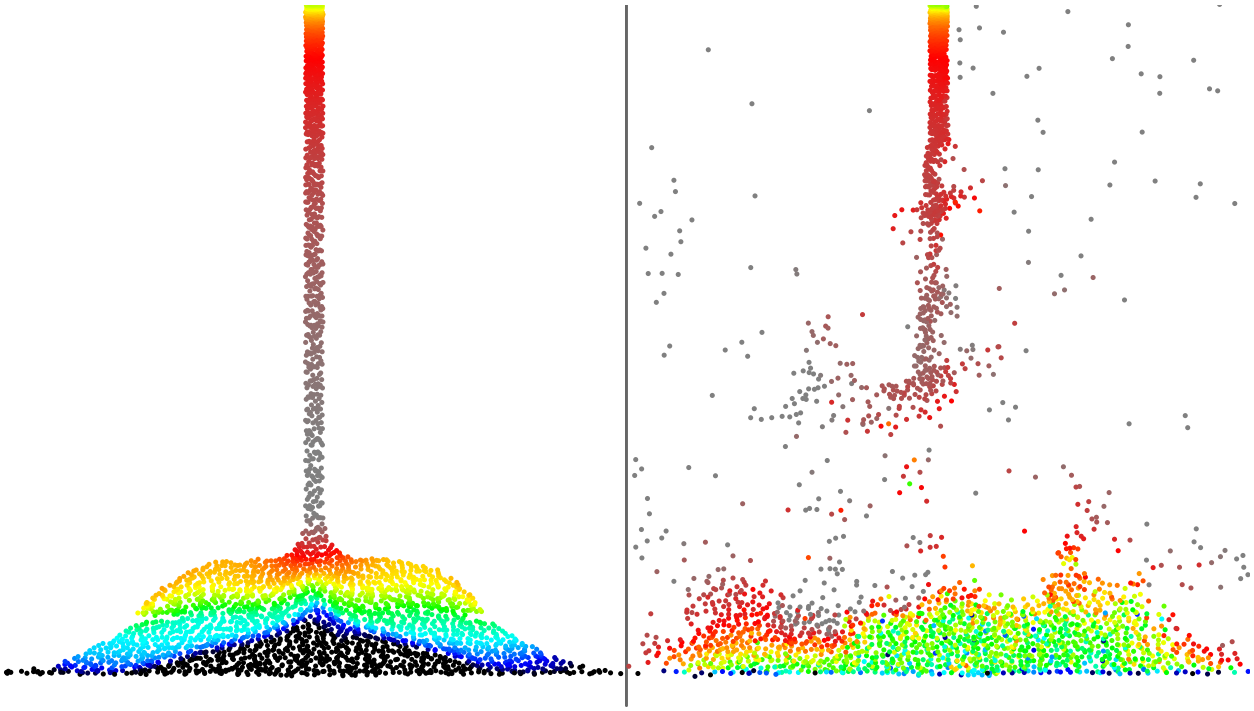


Figure 4.16: *Sand is poured into a pile with APIC (left) and FLIP (right) transfers. FLIP tends to accumulate spurious velocities on particles. In some cases, FLIP leads to unstable behavior, as was the case in this simulation.*

Comparison with the state-of-the-art method. We compare the result of our method with the algorithm proposed by Narain et al. [NGL10] for the collapse of a column of granular material. The results shown in Figure 4.15 are at $150 \times 100 \times 150$ grid resolution. Performance data for this example at a variety of resolutions is shown in Table 4.4. Two particle counts (initial and final) are listed for Narain et al., since their particle counts tend to increase over time due to particle splitting and merging. Although their algorithm runs 8.7 times faster at the highest resolution, our algorithm avoids the stair-casing artifact and is able to produce a less viscous flow of dry cohesion-less granular materials.

4.9 Discussion and limitations

Limitations. There are methods that are much faster, for example the position based dynamics approach in Macklin et al. [MMC14] or other existing continuum approaches such as Narain et al. [NGL10]. However, when realism and intuitively designed parameters are

Grid resolution	Method	Sec/Frame	Particles
$10 \times 15 \times 10$	Ours	0.085	$1.5K$
	Narain	0.019	$1.8 - 2.4K$
$20 \times 30 \times 20$	Ours	1.1	$12K$
	Narain	0.16	$13 - 20K$
$50 \times 75 \times 50$	Ours	33	$188K$
	Narain	3.4	$194 - 330K$
$100 \times 150 \times 100$	Ours	540	$1.5M$
	Narain	62	$1.5 - 2.7M$

Table 4.4: *Performance comparison with Narain et al. [2010].*

more important than raw performance, our method provides an alternative with competitive computational expense. Also, although the framework would generalize to a wide range of yield surfaces and elastic potentials, we only investigated the Drucker-Prager model. However, the Drucker-Prager cone is only equivalent to the Coulomb friction shear/normal-stress relation in two dimensions. In three dimensions, the elastic regime is described by the more complicated region in the Mohr-Coulomb model, but the Drucker-Prager model is a decent approximation [Mas13].

Future work. In future work, we will investigate a wider range of plastic flows and yield surfaces and the effect of cohesion in modeling soil or wet-sands. We would also like to investigate the importance of hardening to visual simulation of sand.

Discussion. We note that explicit time stepping was often faster than implicit time stepping. Although implicit steps are generally larger than explicit, the cost required to solve the nonlinear equations of the implicit step was often larger than just taking more inexpensive explicit steps. Improvements in stability of explicit integration may be partly due to a better position update and APIC transfers providing more stability than FLIP/PIC blends as in Stomakhin et al. [SSC13], whose implicit scheme also benefited from a symmetric treatment. Tuning time step and solver tolerances proved difficult to optimize, requiring different values for different examples.

Acknowledgements

The authors were partially supported by NSF CCF-1422795, ONR (N000141110719, N000141210834), DOD (W81XWH-15-1-0147), Intel STC-Visual Computing Grant (20112360) as well as a gift from Disney Research. We would also like to thank R. Narain for providing invaluable assistance in comparing with their method.

4.10 Derivatives of elasticity and plasticity

As part of an implicit formulation, we encounter the combination

$$\mathbf{Y}(\mathbf{F}) = \frac{\partial \psi}{\partial \mathbf{F}}(\mathbf{Z}(\mathbf{F}, \alpha)) = \mathbf{W}(\mathbf{Z}(\mathbf{F}, \alpha)), \tag{4.32}$$

where $\mathbf{W}(\mathbf{F}) = \frac{\partial \psi}{\partial \mathbf{F}}(\mathbf{F})$. This corresponds to projecting a deformation gradient for plasticity and then using the result as part of a force computation. This function $\mathbf{Y}$ must be differentiated, resulting in the rank-four tensor

$$\mathbf{M} = \frac{\partial \mathbf{Y}}{\partial \mathbf{F}}(\mathbf{F}). \tag{4.33}$$

The tensor $\mathbf{M}$ has $3^4 = 81$ entries and no symmetries. Both the construction and application of $\mathbf{M}$ are somewhat expensive, and both can be avoided.

If $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, then it turns out that $\mathbf{Z}(\mathbf{F}, \alpha) = \mathbf{U}\hat{\mathbf{Z}}(\mathbf{\Sigma}, \alpha)\mathbf{V}^T$ and $\mathbf{W}(\mathbf{F}) = \mathbf{U}\hat{\mathbf{W}}(\mathbf{\Sigma})\mathbf{V}^T$, where $\hat{\mathbf{Z}}(\mathbf{\Sigma}, \alpha)$ and $\hat{\mathbf{W}}(\mathbf{\Sigma})$ are diagonal matrices. It follows then that $\mathbf{Y}(\mathbf{F}) = \mathbf{U}\hat{\mathbf{Y}}(\mathbf{\Sigma})\mathbf{V}^T$, with $\hat{\mathbf{Y}}(\mathbf{\Sigma}) = \hat{\mathbf{W}}(\hat{\mathbf{Z}}(\mathbf{\Sigma}, \alpha))$, where $\hat{\mathbf{Y}}(\mathbf{\Sigma})$ is also a diagonal matrix. To be able to carry out these steps, it is required of the energy density function ψ, that it depends only on the singular values of $\mathbf{F}$. In essence, we need to be able to define $\hat{\psi}$ such that, $\hat{\psi}(\mathbf{\Sigma}) = \psi(\mathbf{F})$. This allows us to write the definition of $\hat{\mathbf{W}}$ as $\hat{\mathbf{W}}(\mathbf{\Sigma}) = \frac{\partial \hat{\psi}}{\partial \mathbf{\Sigma}}(\mathbf{\Sigma})$.

Note that we have taken advantage of these relationships to avoid computing the singular value decomposition more often than necessary. Indeed, $\hat{\mathbf{W}}$ is implemented by ENERGY_DERIVATIVE, and $\hat{\mathbf{Z}}$ is implemented by PROJECT.

Since these functions are rather simple in *diagonal space*, it might not be too surprising that the derivatives are also simpler there. Let $\hat{\mathbf{M}}$ be the diagonal space version of $\mathbf{M}$, defined by

$$M_{ijkl} = \hat{M}_{rsuv} U_{ir} V_{js} U_{ku} V_{lv}, \tag{4.34}$$

where index notation is used and summation is implied. In the pseudocode, the operation

$$\mathbf{A} = \hat{\mathbf{M}} : \mathbf{T} \tag{4.35}$$

is requested. This is equivalent to $A_{ij} = \hat{M}_{ijkl} T_{kl}$. What remains is to determine the structure of $\hat{\mathbf{M}}$. The way that this is done follows from the approach taken in [SHS12], but we summarize the result here.

Introducing the auxiliary variables $\overline{Y}_{ij}, D_{ij}, S_{ij}$, the nonzero entries of $\hat{\mathbf{M}}$ are (with no summation implied)

$$\hat{M}_{iijj} = \overline{Y}_{ij} = \frac{\partial \hat{Y}_{ii}}{\partial \Sigma_{jj}} \tag{4.36}$$

$$\hat{M}_{ijij} = \frac{D_{ij} + S_{ij}}{2} \qquad i \neq j \tag{4.37}$$

$$\hat{M}_{ijji} = \frac{D_{ij} - S_{ij}}{2} \qquad i \neq j \tag{4.38}$$

$$D_{ij} = \frac{\hat{Y}_{ii} - \hat{Y}_{jj}}{\Sigma_{ii} - \Sigma_{jj}} \tag{4.39}$$

$$S_{ij} = \frac{\hat{Y}_{ii} + \hat{Y}_{jj}}{\Sigma_{ii} + \Sigma_{jj}} \tag{4.40}$$

Note that $D_{ij} = D_{ji}$ and $S_{ij} = S_{ji}$, so that $\hat{M}_{ijij} = \hat{M}_{jiji}$ and $\hat{M}_{ijji} = \hat{M}_{jiij}$. Thus, there are only $9+3+3 = 15$ distinct nonzero entries to compute, and $9+6+6 = 21$ multiplications (plus 12 additions) are required to apply the tensor. Of the computations required, $\overline{Y}_{ij}$ and D_{ij} merit further attention. S_{ij} can be computed directly, since division by zero is not a concern there.

First we describe the computation of $\overline{\mathbf{Y}}$. Since $\hat{\mathbf{Y}}$ is the composition of two functions, $\overline{\mathbf{Y}}$

is computed using the chain rule. Note that both $\hat{\mathbf{W}}$ and $\overline{\mathbf{W}}$ are evaluated at $\hat{\mathbf{Z}} = \hat{\mathbf{Z}}(\boldsymbol{\Sigma}, \alpha)$.

$$\overline{Y}_{ij} = \frac{\partial \hat{Y}_{ii}}{\partial \Sigma_{jj}} = \sum_k \frac{\partial \hat{W}_{ii}}{\partial \Sigma_{kk}} \frac{\partial \hat{Z}_{kk}}{\partial \Sigma_{jj}} = \sum_k \overline{W}_{ik} \overline{Z}_{kj}, \tag{4.41}$$

where the matrices $\overline{\mathbf{Y}}$, $\overline{\mathbf{W}}$, and $\overline{\mathbf{Z}}$ represent the derivatives of the functions $\hat{\mathbf{Y}}$, $\hat{\mathbf{W}}$, and $\hat{\mathbf{Z}}$ when diagonal matrices are treated as functions taking vector and returning a vector. Differentiating $\hat{\mathbf{W}}$ gives

$$\overline{\mathbf{W}} = \hat{\mathbf{Z}}^{-1}(2\mu\mathbf{I} - 2\mu \ln(\hat{\mathbf{Z}}) + \lambda \mathbf{o}\mathbf{o}^T - \lambda \operatorname{tr}(\ln(\hat{\mathbf{Z}}))\mathbf{I})\hat{\mathbf{Z}}^{-1}, \tag{4.42}$$

where $\mathbf{o}$ is the all-ones vector.

Next, we need to differentiate the projection to get $\overline{\mathbf{Z}}$. There are three cases to consider. In Case I, $\overline{\mathbf{Z}} = \mathbf{I}$. In Case II, $\overline{\mathbf{Z}} = \mathbf{0}$. This leaves only Case III, in which case

$$\boldsymbol{\varepsilon} = \operatorname{diag}(\ln \boldsymbol{\Sigma}) \quad \mathbf{w} = \operatorname{diag}(\boldsymbol{\Sigma}^{-1}) \quad k = \operatorname{tr}(\ln \boldsymbol{\Sigma}) \quad \mathbf{s} = \boldsymbol{\varepsilon} - \frac{k}{d}\mathbf{o} \quad \hat{\mathbf{s}} = \frac{\mathbf{s}}{\|\mathbf{s}\|} \quad p = \frac{\alpha k(d\lambda + 2\mu)}{2\mu\|\mathbf{s}\|} \tag{4.43}$$

$$\overline{\mathbf{Z}} = \hat{\mathbf{Z}}\left(\left(\frac{1+2p}{d}\mathbf{o} - \frac{p}{k}\boldsymbol{\varepsilon}\right)\mathbf{w}^T - p\left(\mathbf{I} - \hat{\mathbf{s}}\hat{\mathbf{s}}^T\right)\boldsymbol{\Sigma}^{-1}\right). \tag{4.44}$$

With this, $\overline{\mathbf{Y}}$ can be readily computed as $\overline{\mathbf{Y}} = \overline{\mathbf{W}}\,\overline{\mathbf{Z}}$ by Equation (4.41).

Finally, for D_{ij}, we can avoid potential numerical problems in the case where $\Sigma_{ii} \approx \Sigma_{jj}$ by writing

$$D_{ij} = \frac{\hat{Y}_{ii} - \hat{Y}_{jj}}{\Sigma_{ii} - \Sigma_{jj}} = \left(\frac{\hat{Y}_{ii} - \hat{Y}_{jj}}{\hat{Z}_{ii} - \hat{Z}_{jj}}\right)\left(\frac{\hat{Z}_{ii} - \hat{Z}_{jj}}{\Sigma_{ii} - \Sigma_{jj}}\right). \tag{4.45}$$

This works as long as both factors can be robustly computed. Consider the first term.

$$\hat{Y}_{ii} = \frac{2\mu \ln \hat{Z}_{ii}}{\hat{Z}_{ii}} + \frac{\lambda \operatorname{tr}(\ln \hat{\mathbf{Z}})}{\hat{Z}_{ii}} \tag{4.46}$$

$$\hat{Y}_{ii} - \hat{Y}_{jj} = \frac{2\mu(\ln(\hat{Z}_{ii}) - \ln(\hat{Z}_{jj}))}{\hat{Z}_{ii}} - \frac{\lambda \operatorname{tr}(\ln \hat{\mathbf{Z}}) + 2\mu \ln \hat{Z}_{jj}}{\hat{Z}_{ii}\hat{Z}_{jj}}(\hat{Z}_{ii} - \hat{Z}_{jj}) \tag{4.47}$$

$$\frac{\hat{Y}_{ii} - \hat{Y}_{jj}}{\hat{Z}_{ii} - \hat{Z}_{jj}} = \frac{2\mu}{\hat{Z}_{ii}} \frac{\ln(\hat{Z}_{ii}) - \ln(\hat{Z}_{jj})}{\hat{Z}_{ii} - \hat{Z}_{jj}} - \frac{\lambda \operatorname{tr}(\ln \hat{\mathbf{Z}}) + 2\mu \ln \hat{Z}_{jj}}{\hat{Z}_{ii}\hat{Z}_{jj}} \tag{4.48}$$

The only term that presents further difficulties is the divided difference on the natural log. This can be computed by noting

$$\frac{\ln(x) - \ln(y)}{x - y} = \frac{1}{y} \frac{\ln(w+1)}{w} \qquad x = (w+1)y \tag{4.49}$$

$$= \frac{1}{y} \begin{cases} 1 & |w| < \epsilon \\ \operatorname{log1p}(w)/w & |w| \geq \epsilon \end{cases} \tag{4.50}$$

Here we have made use of the log1p library routine, which is designed to be robust in this case.

The next term that must be considered is the divided difference on $\hat{\mathbf{Z}}$. Following the same general procedure yields

$$\frac{\hat{Z}_{ii} - \hat{Z}_{jj}}{\Sigma_{ii} - \Sigma_{jj}} = \left(1 - \frac{\delta\gamma}{\|\hat{\boldsymbol{\varepsilon}}\|_F}\right)\left(\frac{\exp(H_{ii}) - \exp(H_{jj})}{H_{ii} - H_{jj}}\right)\left(\frac{\ln(\Sigma_{ii}) - \ln(\Sigma_{jj})}{\Sigma_{ii} - \Sigma_{jj}}\right). \tag{4.51}$$

The first term is not a problem, and we have already seen how to handle the last term. For the middle term,

$$\frac{e^x - e^y}{x - y} = e^y \frac{e^w - 1}{w} \qquad x = y + w \tag{4.52}$$

$$= e^y \begin{cases} 1 & |w| < \epsilon \\ \operatorname{expm1}(w)/w & |w| \geq \epsilon \end{cases} \tag{4.53}$$

where in this case we have made use of the expm1 library function.

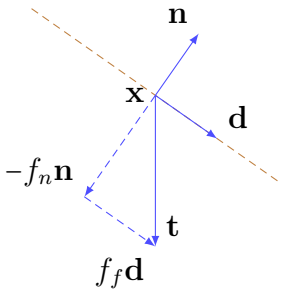
4.11 Yield surface and plastic flow

In the continuum conception of sand, mechanical interactions are expressed through elasticity, modified with plasticity to model the effects of frictional contact. We use the Drucker-Prager plasticity model, which is built to enforce that shear stresses do not exceed a coefficient times normal stresses in magnitude. In Section 4.11.1 we detail the connection between Coulomb friction, and the Drucker-Prager stress condition.

The stress condition defines a notion of admissibility for states of stress. In stress space, this is a region whose boundary is often referred to as the yield surface. This places a constraint on the constitutive model defining the mechanical response of the body. The multiplicative decomposition of the deformation gradient into elastic and plastic parts is a means for designing a constitutive model that meets these constraints. For states of stress in the interior of the feasible region, there is no plastic flow since the elastic constitutive model suffices. However, as a state on the boundary of the region (yield surface) is approached, plastic flow will be defined as means of modifying the constitutive model to satisfy the constraints. In Section 4.12 we derive the plastic flow as a means of satisfying the Drucker-Prager stress constraint.

4.11.1 Drucker-Prager yield surface derivation

Consider a Coulomb friction interaction between two grains in contact. If $\tilde{\alpha}$ is the coefficient of friction, then the frictional force f_f can only be as large as the coefficient of friction times the normal force f_n: $f_f \leq \tilde{\alpha} f_n$. The Drucker-Prager model generalizes this to a continuum. At any point in the continuum body, the Cauchy stress $\boldsymbol{\sigma}$ expresses the local mechanical interactions in the material. Specifically, at point $\mathbf{x}$, $\boldsymbol{\sigma}(\mathbf{x})$ relates the force per area (or traction) $\mathbf{t}$ that material on one side of an imaginary plane with normal $\mathbf{n}$ exerts on material on the other side, as $\mathbf{t} = \boldsymbol{\sigma}(\mathbf{x})\mathbf{n}$. If we consider this interaction to be from friction, we can use the Coulomb model to relate the frictional force (per area) $f_f = \mathbf{d}^T\mathbf{t}$ to the normal force (per area) $f_n = -\mathbf{n}^T\mathbf{t}$ as $\mathbf{d}^T\mathbf{t} \leq -\tilde{\alpha}\mathbf{n}^T\mathbf{t}$.



Here, $\mathbf{d}$ is the normalized projection of the traction $\mathbf{t}$ into the plane orthogonal to $\mathbf{n}$. In terms of $\boldsymbol{\sigma}$, this is expressed as $\mathbf{d}^T\boldsymbol{\sigma}(\mathbf{x})\mathbf{n} \leq -\tilde{\alpha}\mathbf{n}^T\boldsymbol{\sigma}(\mathbf{x})\mathbf{n}$.

The frictional force (per area) $f_f = \mathbf{d}^T\mathbf{t}$ is often referred to as the shear stress (at $\mathbf{x}$, in direction $\mathbf{n}$) and the normal force (per area) is often referred to as the normal stress (at $\mathbf{x}$, in direction $\mathbf{n}$). If we consider all shear stresses to arise from friction, then we get a notion of states of stress consistent with the Coulomb model of frictional interaction. That is, we consider the stress field $\boldsymbol{\sigma}(\mathbf{x})$ as admissible (or consistent with the Coulomb model) if

$$\mathbf{d}^T\boldsymbol{\sigma}(\mathbf{x})\mathbf{n} \leq -\tilde{\alpha}\mathbf{n}^T\boldsymbol{\sigma}(\mathbf{x})\mathbf{n} \tag{4.54}$$

for all $\mathbf{x}$ in the material and for arbitrary directions $\mathbf{d}$ and $\mathbf{n}$ with $\mathbf{d}^T\mathbf{n} = 0$.

When the normal stress $\mathbf{n}^T\boldsymbol{\sigma}(\mathbf{x})\mathbf{n}$ is positive, the material on one side of the imaginary plane is pulling on the material on the other side. This does not arise from a contact/frictional interaction and is a cohesive interaction. Note that Equation (4.54) implies that in the presence of a positive normal stress, the shear stress would have to be zero. In fact, it can be shown that it is not possible to be consistent with Equation (4.54) (for all $\mathbf{d}$ and $\mathbf{n}$) with a positive normal stress, and thus cohesion is not possible with this model.

4.11.1.1 Reformulation of stress admissibility

Consider the two dimensional case and states of stress consistent with Inequality (4.54). In this case, given normal $\mathbf{n}$, there are only two directions $\mathbf{d}$ orthogonal to it, namely $\mathbf{d} = \pm\mathbf{R}\mathbf{n}$ where

$$\mathbf{R} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \tag{4.55}$$

In this case, satisfaction of Inequality (4.54) is achieved when

$$\pm\, \mathbf{n}^T\mathbf{R}\boldsymbol{\sigma}(\mathbf{x})\mathbf{n} + \tilde{\alpha}\mathbf{n}^T\boldsymbol{\sigma}(\mathbf{x})\mathbf{n} \leq 0 \tag{4.56}$$

for all directions $\mathbf{n}$. Since the Cauchy stress must be symmetric (by conservation of angular momentum), it has an eigen decomposition

$$\boldsymbol{\sigma} = \mathbf{Q}\mathbf{D}\mathbf{Q}^T = \mathbf{Q}\begin{pmatrix} s_1 & \\ & s_2 \end{pmatrix}\mathbf{Q}^T \tag{4.57}$$

where $\mathbf{Q}$ is a rotation matrix. Rewriting Inequality (4.56) in terms of the eigen decomposition gives

$$\pm\,\mathbf{n}^T\mathbf{R}\mathbf{Q}\mathbf{D}\mathbf{Q}^T\mathbf{n} + \tilde{\alpha}\mathbf{n}^T\mathbf{Q}\mathbf{D}\mathbf{Q}^T\mathbf{n} \leq 0 \tag{4.58}$$

and since $\mathbf{R}$ and $\mathbf{Q}$ commute (2D rotations commute), satisfaction of Inequality (4.56) is the same as

$$\tilde{\mathbf{n}}^T\left(\pm\mathbf{R}\mathbf{D} + \tilde{\alpha}\mathbf{D}\right)\tilde{\mathbf{n}} \leq 0 \tag{4.59}$$

where $\tilde{\mathbf{n}} = \mathbf{Q}\mathbf{n}$ and

$$\mathbf{R}\mathbf{D} = \begin{pmatrix} & -s_2 \\ s_1 & \end{pmatrix}. \tag{4.60}$$

Since Inequality (4.59) must be true for all $\tilde{\mathbf{n}}$ and choice of sign, it is equivalent to require that the maximum of

$$F(\tilde{\mathbf{n}}, h) = \tilde{\mathbf{n}}^T\left(h\mathbf{R}\mathbf{D} + \tilde{\alpha}\mathbf{D}\right)\tilde{\mathbf{n}} \tag{4.61}$$

subject to $\|\tilde{\mathbf{n}}\|^2 = 1$ and $h^2 = 1$, is less than 0. Using the method of Lagrange multipliers it can be shown that this maximum is given by

$$\frac{s_1 + s_2}{2}\tilde{\alpha} + \frac{|s_1 - s_2|}{2}\sqrt{1 + \tilde{\alpha}^2}. \tag{4.62}$$

Dividing by $\frac{\sqrt{1+\tilde{\alpha}^2}}{\sqrt{2}}$ we obtain that

$$\begin{aligned}(s_1 + s_2)\frac{\tilde{\alpha}}{\sqrt{2}\sqrt{1+\tilde{\alpha}^2}} + \frac{|s_1 - s_2|}{\sqrt{2}} &\leq 0 \\ \operatorname{tr}(\boldsymbol{\sigma}(\mathbf{x}))\alpha + \left\|\boldsymbol{\sigma}(\mathbf{x}) - \frac{\operatorname{tr}(\boldsymbol{\sigma}(\mathbf{x}))}{2}\mathbf{I}\right\|_F &\leq 0\end{aligned} \tag{4.63}$$

Where $\|\cdot\|_F$ is the Frobenius norm and $\alpha = \frac{\tilde{\alpha}}{\sqrt{2}\sqrt{1+\tilde{\alpha}^2}}$.

If we solve the analogous maximization problem in three dimensions we obtain the Mohr-Coulomb yield surface [Mas13]. However, there is a simple generalization of Inequality (4.63) that works for both two and three dimensions given by

$$\operatorname{tr}(\boldsymbol{\sigma}(\mathbf{x}))\alpha + \left\|\boldsymbol{\sigma}(\mathbf{x}) - \frac{\operatorname{tr}(\boldsymbol{\sigma}(\mathbf{x}))}{d}\mathbf{I}\right\|_F \leq 0. \tag{4.64}$$

where d is the number of space dimensions. The Drucker-Prager model uses Inequality (4.64) in both two and three dimensions, because it is easier to work with than the Mohr-Coulomb model in 3D and it is a decent approximation of Mohr-Coulomb in that case.

In summary, the Drucker-Prager model for the stress field $\boldsymbol{\sigma}$ requires that

$$y(\boldsymbol{\sigma}(\mathbf{x})) \leq 0 \tag{4.65}$$

for all points $\mathbf{x}$ in the domain occupied by the material, where $y(\boldsymbol{\sigma}) = \operatorname{tr}(\boldsymbol{\sigma})\alpha + \|\boldsymbol{\sigma} - \frac{\operatorname{tr}(\boldsymbol{\sigma})}{d}\mathbf{I}\|_F$ and d is the number of space dimensions. Note that this function is actually defined in terms of the eigenvalues of $\boldsymbol{\sigma}$ as $y(\boldsymbol{\sigma}) = \operatorname{tr}(\mathbf{D})\alpha + \|\mathbf{D} - \frac{\operatorname{tr}(\mathbf{D})}{d}\mathbf{I}\|_F$.

4.11.2 Kirchhoff stress

The Kirchhoff stress $\boldsymbol{\tau}$ is related to the Cauchy stress $\boldsymbol{\sigma}$ as $\boldsymbol{\tau} = J\boldsymbol{\sigma}$ where $J = \det(\mathbf{F})$ is the determinant of the deformation gradient $\mathbf{F}$. It is often mathematically convenient to express the Drucker-Prager stress condition in terms of this stress measure. We will find this useful when deriving and analyzing properties of the plastic flow. Expressing the Drucker-Prager condition in terms of $\boldsymbol{\tau}$ is simply the requirement that $y(\boldsymbol{\tau}(\mathbf{x})) \leq 0$ for all $\mathbf{x}$ in the domain.

4.11.3 Yield surface

We can think of the condition $y(\boldsymbol{\tau}) = \operatorname{tr}(\boldsymbol{\tau})\alpha + \|\boldsymbol{\tau} - \frac{\operatorname{tr}(\boldsymbol{\tau})}{d}\mathbf{I}\|_F \leq 0$ as defining a feasible region in stress space. Since the constraint can be evaluated as a function of the principal stresses,

we can visualize it as the cone $(\tau_1 + \tau_2)\alpha + \frac{|\tau_1 - \tau_2|}{\sqrt{2}} \leq 0$ for 2D problems, or the cone $(\tau_1 + \tau_2 + \tau_3)\alpha + \sqrt{\sum_{j=1}^{3}\left(\tau_j - \sum_{i=1}^{3}\frac{\tau_i}{3}\right)^2} \leq 0$ for 3D problems. The plastic flow will be chosen as a means of satisfying this constraint. When the stress is in the feasible region, there is no plastic flow. However, when the stress reaches the boundary of this region, the plastic flow will be chosen in a manner that prevents the stress from leaving the feasible region. For this reason, the boundary of the feasible region is called the yield surface, since plastic "yield" occurs when the state of stress reaches it.

4.12 Plastic flow

The plastic flow is characterized by the multiplicative decomposition of the deformation gradient $\mathbf{F} = \mathbf{F}^E\mathbf{F}^P$, however it is convenient for analysis and constitutive modeling to consider evolution of the left elastic Cauchy-Green strain $\mathbf{b}^E = \mathbf{F}^E\mathbf{F}^{E^T} = \mathbf{F}\mathbf{C}^{P^{-1}}\mathbf{F}^T$ where $\mathbf{C}^P = \mathbf{F}^{P^T}\mathbf{F}^P$ is the right plastic Cauchy-Green strain. We will use $\mathbf{l} = \nabla\mathbf{v}$ for brevity throughout. Recalling that the deformation then evolves as $\frac{D\mathbf{F}}{Dt} = \mathbf{l}\mathbf{F}$, $\frac{D\mathbf{b}^E}{Dt} = \mathbf{l}\mathbf{b}^E + \mathbf{b}^E\mathbf{l}^T + \mathbf{F}\frac{D\mathbf{C}^{P^{-1}}}{Dt}\mathbf{F}^T$. The term $\mathbf{F}\frac{D\mathbf{C}^{P^{-1}}}{Dt}\mathbf{F}^T$ is the Lie derivative of the of $\mathbf{b}^E$ with respect to $\mathbf{v}$ so we denote it as $\mathcal{L}_\mathbf{v}\mathbf{b}^E$. The Lie derivative of $\mathbf{b}^E$ is its rate of change independent of deformation in the flow, and it will be determined to define the plastic flow as a means of satisfying the stress feasibility condition in Inequality (4.65). For example, when the stress is inside the feasible region, $\mathcal{L}_\mathbf{v}\mathbf{b}^E = \mathbf{0}$. However, when the stress is on the yield surface, it will be chosen to guarantee that $\dot{y}(t) \leq 0$, thus preventing any future elastic stresses attaining values outside the feasible region. This can be done in infinitely many ways, however care must be taken to avoid artifacts associated with non-volume preserving plastic flows, as well as to guarantee that the plastic flow increases entropy (or decreases the total energy). To illustrate the different choices of $\mathcal{L}_\mathbf{v}\mathbf{b}^E$ in satisfying stress feasibility, we denote it as $\mathcal{L}_\mathbf{v}\mathbf{b}^E = -\gamma\mathbf{L}$ where $\mathbf{L}$ is an arbitrary matrix. With this view, $\mathbf{L}$ is the direction of the Lie derivative and γ is its magnitude. Given any direction $\mathbf{L}$, we can choose magnitude γ to guarantee that $\dot{y}(t) \leq 0$.

4.12.1 Effect of plastic flow on stress criteria

Consider how the stress criteria function $y(\boldsymbol{\tau})$ varies with the elastic state as a function of time: $y(\boldsymbol{\tau}(\mathbf{b}^E(t)))$. The plastic flow will effect this evolution via

$$\begin{aligned}\dot{y}(t) &= \frac{\partial y}{\partial \boldsymbol{\tau}}(\boldsymbol{\tau}(\mathbf{b}^E(t))) : \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{b}^E}(\mathbf{b}^E(t)) : \frac{D\mathbf{b}^E}{Dt}(t) \\ &= \frac{\partial y}{\partial \boldsymbol{\tau}}(\boldsymbol{\tau}(\mathbf{b}^E(t))) : \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{b}^E}(\mathbf{b}^E(t)) : \left(\mathbf{l}\mathbf{b}^E + \mathbf{b}^E\mathbf{l}^T + \mathcal{L}_{\mathbf{v}}\mathbf{b}^E\right)\end{aligned} \tag{4.66}$$

Here, the : operator denotes a generalized dot product to express the chain rule when differentiating the composition of scalar and matrix valued functions of matrix argument. The material derivative $\frac{D}{Dt}$ appears in the chain rule because we are considering how y evolves with time for one particle of the continuum. Defining β as the rate of change of y in the absence of plasticity ($\mathcal{L}_{\mathbf{v}}\mathbf{b}^E = \mathbf{0}$) gives

$$\beta = \frac{\partial y}{\partial \boldsymbol{\tau}}(\boldsymbol{\tau}(\mathbf{b}^E(t))) : \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{b}^E}(\mathbf{b}^E(t)) : \left(\mathbf{l}\mathbf{b}^E + \mathbf{b}^E\mathbf{l}^T\right) \tag{4.67}$$

and using the convention that $\mathcal{L}_{\mathbf{v}}\mathbf{b}^E = -\gamma\mathbf{L}$ gives

$$\dot{y}(t) = \beta - \gamma\frac{\partial y}{\partial \boldsymbol{\tau}}(\boldsymbol{\tau}(\mathbf{b}^E(t))) : \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{b}^E}(\mathbf{b}^E(t)) : \mathbf{L}. \tag{4.68}$$

When the stress criteria is satisfied, we have $y(\boldsymbol{\tau}(\mathbf{b}^E(t))) < 0$ and there is no plastic flow, ($\mathcal{L}_{\mathbf{v}}\mathbf{b}^E = \mathbf{0}$). However, when we reach the boundary of the feasible region in stress space, $y(\boldsymbol{\tau}(\mathbf{b}^E(t))) = 0$, then we will leave the region if $\beta > 0$. In this case, we choose γ so that $\dot{y}(t) = 0$. This defines the plastic flow as

$$\mathcal{L}_{\mathbf{v}}\mathbf{b}^E = \begin{cases} \mathbf{0}, & \text{if } y(\boldsymbol{\tau}(\mathbf{b}^E)) < 0 \text{ or if } y(\boldsymbol{\tau}(\mathbf{b}^E)) = 0 \text{ and } \beta \le 0 \\ -\gamma\mathbf{L}, & \text{if } y(\boldsymbol{\tau}(\mathbf{b}^E)) = 0 \text{ and } \beta > 0 \end{cases} \tag{4.69}$$

where γ is chosen as

$$\gamma = \frac{\beta}{\frac{\partial y}{\partial \boldsymbol{\tau}}(\boldsymbol{\tau}(\mathbf{b}^E(t))) : \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{b}^E}(\mathbf{b}^E(t)) : \mathbf{L}} \tag{4.70}$$

4.12.2 Choosing the direction of the plastic flow

In order to insure that stress never leaves the feasible region, the plastic flow direction $\mathbf{L}$ only needs to have non-zero component $\frac{\partial y}{\partial \boldsymbol{\tau}}(\boldsymbol{\tau}(\mathbf{b}^E(t))) : \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{b}^E}(\mathbf{b}^E(t)) : \mathbf{L}$. Thus, for a given value of $\mathbf{b}^E$, there are infinitely many choices of $\mathbf{L}$ that will suffice in preventing stresses outside the feasible region. However, care must be taken to insure that the plastic flow does not decrease the entropy of the system. Or more specifically, that it does not instantaneously increase the rate of change of the total energy and thus violate the second law of thermodynamics [GS08]. Notably, the rate of change of total energy would be zero in the absence of plasticity so violating this would cause an increase in the total energy. Physically we would expect the plasticity to decrease the total energy over time. We next discuss the choice of $\mathbf{L}$ in light of the entropy concerns.

The total energy $E(t) = KE(t) + PE(t)$ satisfies (see Section 4.14)

$$E(t + \Delta t) - E(t) = W^{\mathbf{t}}(t, \Delta t) - \int_t^{t+\Delta t} \int_{\Omega^0} \dot{w}^P(\mathbf{X}, s) d\mathbf{X} ds \tag{4.71}$$

where $W^{\mathbf{t}}(t, \Delta t)$ is the work done by external traction $\mathbf{t}$ boundary conditions and $\dot{w}^P = \boldsymbol{\tau} : \mathbf{l}^P$ where $\mathbf{l}^P = -\frac{1}{2}\mathcal{L}_{\mathbf{v}}\mathbf{b}^E\mathbf{b}^{E^{-1}}$. In the absence of plasticity, the work done by the mechanical stresses is equal to the negative change in the potential, and this leads to exact conservation of energy (minus the effect of the boundary conditions and external forcing). In the case of plasticity, the total energy may go up or down from the work done by the mechanical stress, and this term quantifies that. Specifically, the plastic flow must be designed in a way that ensures non-negative $\dot{w}^P$, otherwise total energy may increase due to plasticity, which would violate the second law of thermodynamics.

The principle of maximum plastic dissipation [BW08] seeks to design the plastic flow in a way that maximizes $\dot{w}^P$ to respect this concern. This leads to an associative plastic flow where $\mathbf{l}^P = \gamma \frac{\partial y}{\partial \boldsymbol{\tau}}$ or $\mathcal{L}_{\mathbf{v}}\mathbf{b}^E = -2\gamma \frac{\partial y}{\partial \boldsymbol{\tau}}\mathbf{b}^E$. Unfortunately, the choice of matrix $\mathbf{L}$ in $\mathcal{L}_{\mathbf{v}}\mathbf{b}^E = -\gamma\mathbf{L}$ will effect the volume change in the plastic flow. Specifically, it can be shown that if $\mathrm{tr}(\mathbf{L}) = 0$, then the plastic flow will be volume preserving with $J^P = \det(\mathbf{F}^P) = 1$. Since the elastic potential seeks to preserve $\det(\mathbf{F}^E) = 1$ by design, a volume preserving plastic

flow will produce an overall flow that tends to preserve volume. However, without $\text{tr}(\mathbf{L}) = 0$ there is a potential for excessive volume loss or gain in the model and indeed simply using $\mathcal{L}_{\mathbf{v}}\mathbf{b}^E = -2\gamma\frac{\partial y}{\partial \boldsymbol{\tau}}\mathbf{b}^E$ will tend to cause excessive volume gain during sheering [Mas13]. However, using the non-associative rule $\mathcal{L}_{\mathbf{v}}\mathbf{b}^E = -\gamma\mathbf{G}\mathbf{b}^E$, with $\mathbf{G} = \frac{\partial y}{\partial \boldsymbol{\tau}} - \frac{1}{d}\text{tr}(\frac{\partial y}{\partial \boldsymbol{\tau}})\mathbf{I}$, the deviatoric part of $\frac{\partial y}{\partial \boldsymbol{\tau}}$, remedies the artifact. Furthermore, we show in Section 4.14.6 that the modification still guarantees that $\dot{w}^P$ is non-negative and thus satisfies the second law of thermodynamics. In summary, the plasticity is expressed through $\mathcal{L}_{\mathbf{v}}\mathbf{b}^E$ as

$$\mathcal{L}_{\mathbf{v}}\mathbf{b}^E = \begin{cases} \mathbf{0}, & \text{if } y(\boldsymbol{\tau}(\mathbf{b}^E)) < 0 \text{ or if } y(\boldsymbol{\tau}(\mathbf{b}^E)) = 0 \text{ and } \beta \leq 0 \\ -\gamma\mathbf{G}\mathbf{b}^E, \text{ with } \mathbf{G} = \frac{\partial y}{\partial \boldsymbol{\tau}} - \frac{1}{d}\text{tr}(\frac{\partial y}{\partial \boldsymbol{\tau}})\mathbf{I}, & \text{if } y(\boldsymbol{\tau}(\mathbf{b}^E)) = 0 \text{ and } \beta > 0 \end{cases} \tag{4.72}$$

where, given $\mathbf{L} = \mathbf{G}\mathbf{b}^E$, γ is defined as in Equation (4.70).

4.13 Derivation of return mapping algorithm from plastic flow

The return mapping algorithm is the discrete equivalent to solving for a strain that satisfies the plastic flow rule in Equation (4.72) and that lies in the Drucker-Prager yield surface. In this section first we outline the method of Simo and Meschke [SM93] to derive the discrete equations from their continuous versions, and then we show how they can be solved leading to a procedure that computes $\mathbf{Z}(\mathbf{F}^E, \alpha)$. This procedure starts by assuming there is no plastic flow and a return mapping algorithm is derived from the flow equations that shows how to project back to the yield surface if the assumption of no plastic flow is invalid.

Consider the evolution of $\mathbf{b}^E$ from time t^n to time $t^{n+1} = t^n + \Delta t$. We consider this evolution per particle, and thus it is useful to take a Lagrangian view. We outline the notation used in the Lagrangian view in Section 4.14. Specifically useful here is the flow map $\boldsymbol{\phi} : \Omega^0 \times [0, T] \to \mathbb{R}^d$, and its relation to the deformation gradient $\mathbf{F} = \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}}$. Define the time t^n configuration of the material as $\Omega^{t^n} = \{\tilde{\mathbf{x}} | \tilde{\mathbf{x}} = \boldsymbol{\phi}(\mathbf{X}, t^n) \text{ for some } \mathbf{X} \in \Omega^0\}$ and define $\tilde{\boldsymbol{\phi}} : \Omega^{t^n} \times [t^n, T] \to \mathbb{R}^d$ as $\tilde{\boldsymbol{\phi}}(\tilde{\mathbf{x}}, t) = \boldsymbol{\phi}(\boldsymbol{\phi}^{-1}(\tilde{\mathbf{x}}, t^n), t)$. Intuitively, $\tilde{\boldsymbol{\phi}}$ defines the deformation as

if the time t^n configuration Ω^{t^n} of the material is the reference configuration, rather than Ω^0 as in the standard Lagrangian view. This is some times called an updated Lagrangian view. While the deformation gradient $\mathbf{F}$ defines the deformation from the initial configuration (Ω^0) to the time t configuration (Ω^t), the Jacobian $\tilde{\mathbf{F}} = \frac{\partial \tilde{\boldsymbol{\phi}}}{\partial \tilde{\mathbf{x}}}$ defines the deformation from the time t^n configuration (Ω^{t^n}) to the time t configuration (Ω^t), where $t \geq t^n$. Also these are related as $\mathbf{F} = \tilde{\mathbf{F}}\mathbf{F}^n$, or more precisely $\mathbf{F}(\mathbf{X}, t) = \tilde{\mathbf{F}}(\boldsymbol{\phi}(\mathbf{X}, t^n), t)\mathbf{F}(\mathbf{X}, t^n)$ for all $\mathbf{X} \in \Omega^0$.

Define $\mathbf{b}^{E*} = \tilde{\mathbf{F}}^{-1}\mathbf{b}^E\tilde{\mathbf{F}}^{-T}$. Let us consider the difference between the evolution of $\mathbf{b}^{E*}$ and $\mathbf{b}^E$ in absence of plasticity at time $t^n < t < t^{n+1}$. By the definition of $\mathbf{b}^{E*}$, $\frac{D\mathbf{b}^{E*}}{Dt} = -2\gamma\tilde{\mathbf{F}}^{-1}\mathbf{G}\tilde{\mathbf{F}}\mathbf{b}^{E*}$, therefore in absence of plasticity $\mathbf{b}^{E*}$ is constant since $\frac{D\mathbf{b}^{E*}}{Dt} = \mathbf{0}$. In contrast $\mathbf{b}^E$ is not constant in absence of plasticity, as $\mathbf{b}^E\big|_t = \tilde{\mathbf{F}}\big|_t \mathbf{b}^E\big|_{t^n}\tilde{\mathbf{F}}^T\big|_t$. In other words, $\mathbf{b}^{E*}$ is constant along characteristics except for the effect of plasticity, but at the same time $\mathbf{b}^E$ would also be stretched by the flow. This isolation of the plastic part allows for a more intuitive discretization. Specifically, combined with the initial value $\mathbf{b}^{E*}\big|_{t^n} = \mathbf{b}^E\big|_{t^n}$, we can use the exponential approximation $\mathbf{b}^{E*}\big|_{t^{n+1}} \approx \exp(-2\delta\gamma\,\tilde{\mathbf{F}}^{-1}\mathbf{G}\tilde{\mathbf{F}})\big|_{t^{n+1}}\mathbf{b}^E\big|_{t^n}$ where $\delta\gamma \geq 0$ will be used to enforce the constraint $y(\boldsymbol{\tau}(\mathbf{b}^E\big|_{t^{n+1}})) \leq 0$. Multiplying the approximation by $\tilde{\mathbf{F}}\big|_{t^{n+1}}$ on the left and $\tilde{\mathbf{F}}^T\big|_{t^{n+1}}$ on the right, and recalling the definition of $\mathbf{b}^{E*}$, we obtain

$$
\begin{aligned}
\mathbf{b}^E\big|_{t^{n+1}} &= \tilde{\mathbf{F}}\big|_{t^{n+1}}\mathbf{b}^{E*}\big|_{t^{n+1}}\tilde{\mathbf{F}}^T\big|_{t^{n+1}} \\
&\approx \tilde{\mathbf{F}}\big|_{t^{n+1}}\exp(-2\delta\gamma\,\tilde{\mathbf{F}}^{-1}\mathbf{G}\tilde{\mathbf{F}})\big|_{t^{n+1}}\mathbf{b}^E\big|_{t^n}\tilde{\mathbf{F}}^T\big|_{t^{n+1}} \\
&= \tilde{\mathbf{F}}\big|_{t^{n+1}}\tilde{\mathbf{F}}^{-1}\big|_{t^{n+1}}\exp(-2\delta\gamma\,\mathbf{G})|_{t^{n+1}}\tilde{\mathbf{F}}\big|_{t^{n+1}}\mathbf{b}^E\big|_{t^n}\tilde{\mathbf{F}}^T\big|_{t^{n+1}} \\
&= \exp(-2\delta\gamma\,\mathbf{G})|_{t^{n+1}}\tilde{\mathbf{F}}\big|_{t^{n+1}}\mathbf{b}^E\big|_{t^n}\tilde{\mathbf{F}}^T\big|_{t^{n+1}}.
\end{aligned}
$$

Using the notation $\hat{\mathbf{B}}^E = \tilde{\mathbf{F}}\big|_{t^{n+1}}\mathbf{b}^E\big|_{t^n}\tilde{\mathbf{F}}^T\big|_{t^{n+1}}$, we are looking for a solution pair $\delta\gamma$ and $\mathbf{b}^E\big|_{t^{n+1}}$ such that

$$
\mathbf{b}^E\big|_{t^{n+1}} = \exp\left(-2\delta\gamma\,\mathbf{G}(\boldsymbol{\tau}(\mathbf{b}^E\big|_{t^{n+1}}))\right)\hat{\mathbf{B}}^E, \tag{4.73}
$$

and constraint $y(\boldsymbol{\tau}(\mathbf{b}^E\big|_{t^{n+1}})) \leq 0$ is satisfied. Note that $\hat{\mathbf{B}}^E$ is the elastic strain we would get without the effect of plasticity. For example if $y(\boldsymbol{\tau}(\hat{\mathbf{B}}^E)) \leq 0$, then $\delta\gamma = 0$ and $\mathbf{b}^E\big|_{t^{n+1}} = \hat{\mathbf{B}}^E$

is the trivial solution pair and there is no plastic flow. In this sense, we can see that $\hat{\mathbf{B}}^E$ can be considered as the trial elastic state obtained without any plastic flow. If this does not satisfy the constraint, $\delta\gamma$ and $\mathbf{b}^E|_{t^{n+1}}$ must be defined to "project" $\hat{\mathbf{B}}^E$ to $\mathbf{b}^E|_{t^{n+1}}$.

We use this process to define the projection $\mathbf{Z}(\mathbf{F}^E, \alpha)$. $\mathbf{F}^E$ is considered the trial elastic state, one obtained in the absence of plastic flow. Thus, $\hat{\mathbf{B}}^E = \mathbf{F}^E\mathbf{F}^{E^T}$ and we seek the solution of Equation 4.73 to define the projection to $\mathbf{b}^E|_{t^{n+1}}$, from which we can determine $\mathbf{Z}(\mathbf{F}^E, \alpha)$. This can be done most easily by considering the singular value decomposition of $\mathbf{F}^E$.

If the singular value decomposition of $\mathbf{F}^E$ is given by $\mathbf{F}^E = \mathbf{U}^E\mathbf{\Sigma}^E\mathbf{V}^{E^T}$, then $\hat{\mathbf{B}}^E = \mathbf{F}^E\mathbf{F}^{E^T} = \mathbf{U}^E\mathbf{\Sigma}^{E^2}\mathbf{U}^{E^T}$. It can be shown that $\mathbf{U}$ diagonalizes $\mathbf{G}(\boldsymbol{\tau}(\mathbf{b}^E|_{t^{n+1}}))$ and $\mathbf{b}^E|_{t^{n+1}}$ (i.e. $\mathbf{G}(\boldsymbol{\tau}(\mathbf{b}^E|_{t^{n+1}})) = \mathbf{U}^E\hat{\mathbf{G}}(\mathbf{\Sigma}^{E,n+1})\mathbf{U}^{E^T}$, and $\mathbf{b}^E|_{t^{n+1}} = \mathbf{U}^E\left(\mathbf{\Sigma}^{E,n+1}\right)^2\mathbf{U}^{E^T}$), then we may write (4.73) as

$$\mathbf{U}^E\left(\mathbf{\Sigma}^{E,n+1}\right)^2\mathbf{U}^{E^T} = \exp\left(-2\delta\gamma\mathbf{U}^E\hat{\mathbf{G}}(\mathbf{\Sigma}^{E,n+1})\mathbf{U}^{E^T}\right)\mathbf{U}^E\mathbf{\Sigma}^{E^2}\mathbf{U}^{E^T} = \mathbf{U}^E\exp\left(-2\delta\gamma\hat{\mathbf{G}}(\mathbf{\Sigma}^{E,n+1})\right)\mathbf{\Sigma}^{E^2}\mathbf{U}^{E^T}. \tag{4.74}$$

Multiplying both sides of Equation (4.74) by $\mathbf{U}^{E^T}$ on the left and by $\mathbf{U}^E$ on the right, and taking log results in

$$2\ln\left(\mathbf{\Sigma}^{E,n+1}\right) = -2\delta\gamma\hat{\mathbf{G}}(\mathbf{\Sigma}^E) + 2\ln\left(\mathbf{\Sigma}^E\right). \tag{4.75}$$

The model that we choose uses the Hencky-strain as a measure of deformation. By defining

$$\boldsymbol{\varepsilon}^E := \ln\mathbf{\Sigma}^E \quad \text{and} \quad \mathbf{H}^E := \ln\mathbf{\Sigma}^{E,n+1}, \tag{4.76}$$

we may simplify and rearrange Equation (4.75)

$$\boldsymbol{\varepsilon}^E - \mathbf{H}^E = \delta\gamma\hat{\mathbf{G}}. \tag{4.77}$$

This is our discrete flow rule. In the return mapping algorithm, we want to solve for $\mathbf{H}^E$

satisfies Equation (4.77) subject to the constraint

$$y(\boldsymbol{\tau}(\mathbf{H}^E)) \leq 0. \tag{4.78}$$

Solving Equation (4.77) and (4.78) can be seen as a ray-cone intersection problem, see Figure 9 in the paper. Before proceeding, we introduce the deviatoric operator to act on matrices:

$$\operatorname{dev}(\mathbf{A}) \coloneqq \mathbf{A} - \frac{1}{d}\operatorname{tr}(\mathbf{A})\mathbf{I}, \tag{4.79}$$

i.e. $\operatorname{dev}(\mathbf{A})$ gives the deviatoric part of any arbitrary square matrix $\mathbf{A}$ of size $d \times d$. Equation (4.77) has no solution if $\operatorname{tr}(\boldsymbol{\varepsilon}^E) \geq 0$. In this case the sand is in extension and we project to the tip $\mathbf{H}^E = \mathbf{0}$. We have $\mathbf{G} = \operatorname{dev}(\frac{\partial y}{\partial \boldsymbol{\tau}})$, and $\frac{\partial y}{\partial \boldsymbol{\tau}} = \alpha\mathbf{I} + \frac{\operatorname{dev}(\boldsymbol{\tau})}{\|\operatorname{dev}(\boldsymbol{\tau})\|_F}$, thus $\mathbf{G}$ is simply $\frac{\operatorname{dev}(\boldsymbol{\tau})}{\|\operatorname{dev}(\boldsymbol{\tau})\|_F}$. In principal space this becomes $\hat{\mathbf{G}} = \frac{\operatorname{dev}(\hat{\boldsymbol{\tau}})}{\|\operatorname{dev}(\hat{\boldsymbol{\tau}})\|_F}$, where $\hat{\boldsymbol{\tau}}$ and $\hat{\mathbf{G}}$ are diagonal. From (§4.14.5) we have $\hat{\boldsymbol{\tau}} = \frac{\partial \psi}{\partial \boldsymbol{\varepsilon}^E} = 2\mu\mathbf{H}^E + \lambda \operatorname{tr}(\mathbf{H}^E)\mathbf{I}$ because we use the energy density $\psi(\boldsymbol{\varepsilon}^E) = \mu \operatorname{tr}((\boldsymbol{\varepsilon}^E)^2) + \frac{1}{2}\lambda \operatorname{tr}(\boldsymbol{\varepsilon}^E)^2$. Thus $\hat{\mathbf{G}} = \frac{\operatorname{dev}(\mathbf{H}^E)}{\|\operatorname{dev}(\mathbf{H}^E)\|_F}$. Using Equation (4.77), we can see that $\operatorname{tr}(\boldsymbol{\varepsilon}) = \operatorname{tr}(\mathbf{H}^E)$, since $\operatorname{tr}(\hat{\mathbf{G}}) = 0$. Thus $\operatorname{dev}(\boldsymbol{\varepsilon}^E) - \operatorname{dev}(\mathbf{H}^E) = \delta\gamma \frac{\operatorname{dev}(\mathbf{H}^E)}{\|\operatorname{dev}(\mathbf{H}^E)\|_F}$, and collecting like terms we have $\operatorname{dev}(\boldsymbol{\varepsilon}^E) = \left(1 + \frac{\delta\gamma}{\|\operatorname{dev}(\mathbf{H}^E)\|_F}\right)\operatorname{dev}(\mathbf{H}^E)$. Thus $\hat{\mathbf{G}} = \frac{\operatorname{dev}(\boldsymbol{\varepsilon}^E)}{\|\operatorname{dev}(\boldsymbol{\varepsilon}^E)\|_F}$. Then plugging the equation for the ray $\mathbf{H}^E = \boldsymbol{\varepsilon}^E - \delta\gamma \frac{\operatorname{dev}(\boldsymbol{\varepsilon}^E)}{\|\operatorname{dev}(\boldsymbol{\varepsilon}^E)\|_F}$, into the equation for the cone $y(\boldsymbol{\tau}(\mathbf{H}^E)) = \mathbf{0}$, and solving for $\delta\gamma$, we obtain

$$\delta\gamma = \|\operatorname{dev}(\boldsymbol{\varepsilon}^E)\|_F + \left(\frac{d\lambda + 2\mu}{2\mu}\right)\operatorname{tr}(\boldsymbol{\varepsilon}^E)\alpha. \tag{4.80}$$

If $\delta\gamma \leq 0$ we intersect the cone from the inside and thus don't need to project and have $\mathbf{H}^E = \boldsymbol{\varepsilon}^E$. Otherwise we project to the cone and $\mathbf{H}^E = \boldsymbol{\varepsilon}^E - \delta\gamma\hat{\boldsymbol{\varepsilon}}^E$. Finally, we return $\mathbf{Z}(\mathbf{F}^E, \alpha) = \mathbf{U}e^{\mathbf{H}^E}\mathbf{V}^T$.

4.14 Energy and plasticity

Here we discuss the notion of total, kinetic and potential energy in the context of elastoplasticity. It is important to carefully consider the effect the plastic flow will have on the rate

of change of total energy. The plastic flow should not increase the rate of change of total energy. Using a hyperelastic constitutive model for the elastic stress implies that the rate of change of total energy in the absence of plasticity will be zero. We take a Lagrangian view of the continuum for these derivations. We define a number of quantities here for completeness but refer the reader to the texts of Gonzalez and Stuart [GS08] and Bonet and Wood [BW08] for more detail on Lagrangian and Eulerian descriptions of the continuum.

We use $\boldsymbol{\phi} : \Omega^0 \times [0,T] \to \mathbb{R}^d$ to denote the flow map of the material (where $d = 2$ or 3 is the number of space dimensions). The Lagrangian view identifies particles of the continuum with their initial positions. Ω^0 is the set of all initial positions of particles in the material. We use $\mathbf{X}$ to represent points in Ω^0. We use $\Omega^t = \{\mathbf{x} |\ \mathbf{x} = \boldsymbol{\phi}(\mathbf{X}, t) \text{ for some } \mathbf{X} \in \Omega^0\}$ to represent the time t configuration of the material. In other words, $\boldsymbol{\phi}(\cdot, t) : \Omega^0 \to \Omega^t$ and $\boldsymbol{\phi}(\mathbf{X}, t)$ is the location of particle $\mathbf{X}$ at time t. Thus $\boldsymbol{\phi}(\mathbf{X}, t)$ is the trajectory of the material point $\mathbf{X}$ over time, and $\mathbf{V}(\mathbf{X}, t) = \frac{\partial \boldsymbol{\phi}}{\partial t}(\mathbf{X}, t)$ is its velocity and $\mathbf{A}(\mathbf{X}, t) = \frac{\partial^2 \boldsymbol{\phi}}{\partial t^2}(\mathbf{X}, t)$ is its acceleration. Note also that the deformation gradient is related to the flow map as $\mathbf{F} = \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{X}}$. The flow map is invertible (a fundamental assumption of continuum mechanics) and its inverse $\boldsymbol{\phi}^{-1}$ can be used to define any function over Ω^0 as a function over Ω^t. For example, the Eulerian velocity is related to the flow map as $\mathbf{v}(\mathbf{x}, t) = \mathbf{V}(\boldsymbol{\phi}^{-1}(\mathbf{x}, t), t)$. We can also define a Lagrangian version of the mass density $R : \Omega^0 \times [0,T] \to \mathbb{R}$ with $R(\mathbf{X}, t) = \rho(\phi(\mathbf{X}, t), t)$ and $\rho(\mathbf{x}, t) = R(\boldsymbol{\phi}^{-1}(\mathbf{x}, t), t)$.

The Lagrangian view presents more options when defining stresses, for example while the Cauchy stress ($\boldsymbol{\sigma}$) relates area weighted normals in the current configuration (Ω^t) to surface tractions, the first Piola-Kirchhoff Stress ($\mathbf{P}$) relates area weighted normals in the initial configuration (Ω^0) to surface tractions. We will use the first Piola-Kirchhoff stress tensor in our discussion of energy.

We can also express the governing equations in the Lagrangian view. Conservation of mass, in the Lagrangian view is

$$R(\mathbf{X}, t) J(\mathbf{X}, t) = R(\mathbf{X}, 0), \quad \mathbf{X} \in \Omega^0,\ t \in [0,T] \tag{4.81}$$

where recall that $J(\mathbf{X},t) = \det(\mathbf{F}(\mathbf{X},t))$. Conservation of linear momentum results in force density balance

$$R(\mathbf{X},0)\mathbf{A}(\mathbf{X},t) = \nabla^{\mathbf{X}} \cdot \mathbf{P}(\mathbf{X},t), \quad \mathbf{X} \in \Omega^0,\ t \in [0,T]. \tag{4.82}$$

Note that this equation has units of force density but is otherwise just Newton's second law generalized to the continuum.

4.14.1 Work done by elastic deformation

The work done by the elastic forces (W^e) is defined to be (see [BW08])

$$W^e(T) = \int_0^T \int_{\Omega^0} P_{ij,j} V_i d\mathbf{X} dt. \tag{4.83}$$

Here we use the convention that $P_{ij,jk}$ represents $\frac{\partial P_{ij}}{\partial X_k}$ and unless otherwise stated, we use the Einstein summation conventions where repeated indices are summed over their ranges. With this in mind, the work can be rewritten by using integration by parts, this satisfies

$$\int_0^T \int_{\Omega^0} P_{ij,j} V_i d\mathbf{X} dt = \int_0^T \int_{\Omega^0} (P_{ij}V_i)_{,j} - P_{ij}V_{i,j} d\mathbf{X} dt = \int_0^T \int_{\partial\Omega^0} t_i V_i dS(\mathbf{X}) - \int_{\Omega^0} P_{ij}V_{i,j} d\mathbf{X} dt \tag{4.84}$$

where $t_i = P_{ij}N_j$ is the applied traction boundary condition. Now since

$$\frac{d}{dt}\int_{\Omega^0} \psi(\mathbf{F}(\mathbf{X},t)) d\mathbf{X} = \int_{\Omega^0} \frac{d\psi}{dF_{ij}}(\mathbf{F}(\mathbf{X},t)) V_{i,j}(\mathbf{X},t) d\mathbf{X} = \int_{\Omega^0} P_{ij}V_{i,j} d\mathbf{X} \tag{4.85}$$

we can say

$$\begin{aligned} W^e(T) &= W^{\mathbf{t}}(T) - \int_0^T \frac{d}{dt}\int_{\Omega^0} \psi(\mathbf{F}(\mathbf{X},t)) d\mathbf{X} dt \\ &= W^{\mathbf{t}}(T) - \int_{\Omega^0} \psi(\mathbf{F}(\mathbf{X},T)) d\mathbf{X} + \int_{\Omega^0} \psi(\mathbf{F}(\mathbf{X},0)) d\mathbf{X} \\ &= W^{\mathbf{t}}(T) - PE(T) + PE(0) \end{aligned}$$

where $W^t(T) = \int_0^T \int_{\partial\Omega^0} t_i V_i dS(\mathbf{X}) dt$ is defined to be the work done by the boundary forces and $PE(t)$ is the elastic potential at time t. The kinetic energy $KE(t)$ is

$$KE(t) = \int_{\Omega_0} \frac{1}{2} \mathbf{V}(\mathbf{X}, t)^T \big(R(\mathbf{X}, 0) \mathbf{V}(\mathbf{X}, t) \big) d\mathbf{X} \tag{4.86}$$

The rate of change of kinetic energy density is:

$$\frac{d}{dt} \left[\frac{1}{2} \mathbf{V}(\mathbf{X}, t)^T \big(R(\mathbf{X}, 0) \mathbf{V}(\mathbf{X}, t) \big) \right] = R(\mathbf{X}, 0) \mathbf{A}(\mathbf{X}, t) \cdot \mathbf{V}(\mathbf{X}, t) \tag{4.87}$$

and assuming

$$R(\mathbf{X}, 0) \mathbf{A}(\mathbf{X}, t) = \nabla^{\mathbf{X}} \cdot \mathbf{P} \tag{4.88}$$

we get

$$\int_0^T \int_{\Omega_0} \frac{d}{dt} \left[\frac{1}{2} \mathbf{V}(\mathbf{X}, t)^T \big(R(\mathbf{X}, 0) \mathbf{V}(\mathbf{X}, t) \big) \right] d\mathbf{X} dt = KE(T) - KE(0) = W^e \tag{4.89}$$

and also

$$KE(T) - KE(0) + PE(T) - PE(0) = W^{\mathbf{t}}(T) \tag{4.90}$$

4.14.2 Plastic flow rate and potential

With plasticity we have $\mathbf{F} = \mathbf{F}^E \mathbf{F}^P$ and $\dot{\mathbf{F}} = \dot{\mathbf{F}}^E \mathbf{F}^P + \mathbf{F}^E \dot{\mathbf{F}}^P$. Also,

$$\dot{\mathbf{F}}^E = \dot{\mathbf{F}} \mathbf{F}^{P^{-1}} - \mathbf{F}^E \dot{\mathbf{F}}^P \mathbf{F}^{P^{-1}} \tag{4.91}$$

etc. The elastic potential energy is defined as

$$PE(t) = \int_{\Omega_0} \psi(\mathbf{F}^E(\mathbf{X}, t)) d\mathbf{X} \tag{4.92}$$

and its rate of change is

$$\begin{aligned}\frac{d}{dt}PE(t) &= \int_{\Omega_0} \frac{\partial \psi}{\partial F_{ij}}(\mathbf{F}^E(\mathbf{X},t))\dot{F}^E_{ij}(\mathbf{X},t)d\mathbf{X} \\ &= \int_{\Omega_0} \frac{\partial \psi}{\partial F_{ij}}(\mathbf{F}^E)F^{P^{-T}}_{jk}\dot{F}_{ik} - F^{E^T}_{ki}\frac{\partial \psi}{\partial F_{ij}}(\mathbf{F}^E)F^{P^{-T}}_{jl}\dot{F}^P_{kl}d\mathbf{X}.\end{aligned}$$

The integral above motivates the definition of the first Piola-Kirchhoff Stress in the presence of plasticity as $P_{ik} = \frac{\partial \psi}{\partial F_{ij}}(\mathbf{F}^E)F^{P^{-T}}_{jk}$ or

$$\mathbf{P}(\mathbf{X},t) = \frac{\partial \psi}{\partial \mathbf{F}}(\mathbf{F}^E(\mathbf{X},t))\mathbf{F}^{P^{-T}}(\mathbf{X},t). \tag{4.93}$$

With this definition, the work done by the mechanical forces is

$$\begin{aligned}W^e(T) &= \int_0^T \int_{\Omega_0} P_{ij,j}(\mathbf{X},t)V_i(\mathbf{X},t)d\mathbf{X}dt \\ &= W^{\mathbf{t}}(T) - \int_0^T \int_{\Omega_0} \frac{\partial \psi}{\partial F_{ij}}(\mathbf{F}^E)\dot{F}_{ik}F^{P^{-T}}_{jk}d\mathbf{X}dt\end{aligned}$$

4.14.3 Stress power density

Define the stress power density as

$$\dot{w}(\mathbf{X},t) = \tau_{ij}(\mathbf{X},t)l_{ij}(\mathbf{X},t) = P_{ij}(\mathbf{X},t)\dot{F}_{ij}(\mathbf{X},t) \tag{4.94}$$

with $\mathbf{l} = \dot{\mathbf{F}}\mathbf{F}^{-1}$ and $\boldsymbol{\tau} = J\boldsymbol{\sigma} = \mathbf{P}\mathbf{F}^T$. Also define

$$\dot{w}^e(\mathbf{X},t) = \tau_{ij}(\mathbf{X},t)l^e_{ij}(\mathbf{X},t) = \frac{\partial \psi}{\partial F_{ij}}(\mathbf{F}^E(\mathbf{X},t))\dot{F}^E_{ij}(\mathbf{X},t) \tag{4.95}$$

with $\mathbf{l}^E = \dot{\mathbf{F}}^E\mathbf{F}^{E^{-1}}$. $\dot{w}^e(\mathbf{X},t)$ is then the rate of change in elastic potential density since

$$\frac{d}{dt}PE(t) = \int_{\Omega_0} \frac{\partial \psi}{\partial F_{ij}}(\mathbf{F}^E(\mathbf{X},t))\dot{F}^E_{ij}(\mathbf{X},t)d\mathbf{X} = \int_{\Omega_0} \dot{w}^e(\mathbf{X},t)d\mathbf{X} \tag{4.96}$$

Next, defining

$$\dot{w}^p(\mathbf{X},t) = F^{E^T}_{ki}\frac{\partial \psi}{\partial F_{ij}}(\mathbf{F}^E)F^{P-T}_{jl}\dot{F}^P_{kl} = P_{il}F^E_{ik}\dot{F}^P_{kl} \tag{4.97}$$

gives

$$\dot{w}(\mathbf{X},t) = \dot{w}^e(\mathbf{X},t) + \dot{w}^p(\mathbf{X},t). \tag{4.98}$$

The term $\mathbf{F}^E\dot{\mathbf{F}^P}$ is referred to as the plastic rate of deformation [BW08]. The work can then expressed as

$$W^e(T) = \int_0^T \int_{\partial\Omega^0} t_i V_i dS(\mathbf{X}) - \int_{\Omega^0} P_{ij}V_{i,j}d\mathbf{X}dt = W^{\mathbf{t}}(T) - \int_0^T \int_{\Omega_0} \dot{w}^e(\mathbf{X},t) + \dot{w}^p(\mathbf{X},t)d\mathbf{X}dt \tag{4.99}$$

and then also

$$\begin{aligned} KE(T) - KE(0) &= W^{\mathbf{t}}(T) - \int_0^T \int_{\Omega_0} \dot{w}^e(\mathbf{X},t) + \dot{w}^p(\mathbf{X},t)d\mathbf{X}dt \\ &= W^{\mathbf{t}}(T) - PE(T) + PE(0) - \int_0^T \int_{\Omega_0} \dot{w}^p(\mathbf{X},t)d\mathbf{X}dt \end{aligned}$$

and thus

$$KE(T) - KE(0) + PE(T) - PE(0) = W^{\mathbf{t}}(T) - \int_0^T \int_{\Omega_0} \dot{w}^p(\mathbf{X},t)d\mathbf{X}dt. \tag{4.100}$$

This motivates why $\dot{w}^p(\mathbf{X},t)$ is often referred to as "plastic dissipation rate".

4.14.4 Hencky strain derivative lemma

Consider symmetric positive definite matrix $\mathbf{B} \in \mathbb{R}^{d\times d}$ with $d = 2$ or 3. Use $\mathbf{B} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ to denote the eigenvalue decomposition of $\mathbf{B}$ where $\mathbf{\Lambda}$ is diagonal and positive definite. Define $\boldsymbol{\varepsilon}(\mathbf{B}) = \frac{1}{2}\ln(\mathbf{B}) = \mathbf{U}\ln(\mathbf{\Lambda}^{\frac{1}{2}})\mathbf{U}^T$. For example if $\mathbf{B} = \mathbf{F}\mathbf{F}^T$, then $\boldsymbol{\varepsilon}$ is the Hencky strain. We can also write $\mathbf{B}(\boldsymbol{\varepsilon}) = e^{2\boldsymbol{\varepsilon}}$.

Lemma: Suppose that $f(\mathbf{B})$ is a scalar function of $\mathbf{B}$ which is invariant under coordinate changes, and $\hat{f}(\boldsymbol{\varepsilon}) = f(\mathbf{B}(\boldsymbol{\varepsilon}))$, then $\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}} = 2\frac{\partial f}{\partial \mathbf{B}}\mathbf{B}$.

Proof: First note that $\hat{f}$ will also be invariant under coordinate change and therefore

can be written as a function of the invariants of $\boldsymbol{\varepsilon}$. That is $\hat{f}(\boldsymbol{\varepsilon}) = \tilde{f}(I_1, I_2, I_3)$. This means we have $\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}} = \frac{\partial \tilde{f}}{\partial I_1}\frac{\partial I_1}{\partial \boldsymbol{\varepsilon}} + \frac{\partial \tilde{f}}{\partial I_2}\frac{\partial I_2}{\partial \boldsymbol{\varepsilon}} + \frac{\partial \tilde{f}}{\partial J}\frac{\partial J}{\partial \boldsymbol{\varepsilon}}$ where $\frac{\partial I_1}{\partial \boldsymbol{\varepsilon}} = \mathbf{I}$, $\frac{\partial I_2}{\partial \boldsymbol{\varepsilon}} = I_1\mathbf{I} - \boldsymbol{\varepsilon}^T$, $\frac{\partial J}{\partial \boldsymbol{\varepsilon}} = J\boldsymbol{\varepsilon}^{-T}$. Therefore, if $\boldsymbol{\varepsilon}$ is diagonal then $\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}$ will be as well. Note that from frame invariance we have $\hat{f}(\boldsymbol{\varepsilon}) = \hat{f}(\mathbf{R}^T\boldsymbol{\varepsilon}\mathbf{R})$ for any rotation $\mathbf{R}$. Which means

$$
\begin{aligned}
\left.\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\right|_{\boldsymbol{\varepsilon}} : \delta\boldsymbol{\varepsilon} &= \left.\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\right|_{\mathbf{R}^T\boldsymbol{\varepsilon}\mathbf{R}} : \mathbf{R}^T\delta\boldsymbol{\varepsilon}\mathbf{R} \\
\left.\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\right|_{\boldsymbol{\varepsilon}} : \delta\boldsymbol{\varepsilon} &= \mathbf{R}\left.\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\right|_{\mathbf{R}^T\boldsymbol{\varepsilon}\mathbf{R}} \mathbf{R}^T : \delta\boldsymbol{\varepsilon} \\
\left.\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\right|_{\boldsymbol{\varepsilon}} &= \mathbf{R}\left.\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\right|_{\mathbf{R}^T\boldsymbol{\varepsilon}\mathbf{R}} \mathbf{R}^T \\
\mathbf{R}\left.\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\right|_{\boldsymbol{\varepsilon}} \mathbf{R}^T &= \left.\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\right|_{\mathbf{R}^T\boldsymbol{\varepsilon}\mathbf{R}}
\end{aligned}
$$

Plugging in $\mathbf{R} = \mathbf{U}$ we have $\mathbf{U}\left.\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\right|_{\boldsymbol{\varepsilon}} \mathbf{U}^T = \left.\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\right|_{\mathbf{U}^T\boldsymbol{\varepsilon}\mathbf{U}}$ and is therefore diagonal.

Deriving

$$
\begin{aligned}
\frac{\partial f}{\partial \mathbf{B}} : \delta\mathbf{B} &= \frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}} : \delta\boldsymbol{\varepsilon} \\
&= \mathbf{U}^T\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\mathbf{U} : \mathbf{U}^T\delta\boldsymbol{\varepsilon}\mathbf{U} \\
&= \mathbf{U}^T\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\mathbf{U} : \operatorname{diag}(\mathbf{U}^T\delta\boldsymbol{\varepsilon}\mathbf{U})
\end{aligned}
$$

From $\boldsymbol{\varepsilon} = \frac{1}{2}\mathbf{U}\log(\boldsymbol{\Lambda})\mathbf{U}^T$ we have

$$
\begin{aligned}
\delta\boldsymbol{\varepsilon} &= \frac{1}{2}(\delta\mathbf{U}\log(\boldsymbol{\Lambda})\mathbf{U}^T + \mathbf{U}\delta\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{-1}\mathbf{U}^T + \mathbf{U}\log(\boldsymbol{\Lambda})\delta\mathbf{U}^T) \\
\mathbf{U}^T\delta\boldsymbol{\varepsilon}\mathbf{U} &= \frac{1}{2}(\mathbf{U}^T\delta\mathbf{U}\log(\boldsymbol{\Lambda}) + \delta\boldsymbol{\Lambda}\boldsymbol{\Lambda}^{-1} + \log(\boldsymbol{\Lambda})\delta\mathbf{U}^T\mathbf{U})
\end{aligned}
$$

Since $\mathbf{U}$ is orthonormal $\mathbf{U}^T\delta\mathbf{U}$ is skew and therefore $\operatorname{diag}(\mathbf{U}^T\delta\boldsymbol{\varepsilon}\mathbf{U}) = \operatorname{diag}(\delta\boldsymbol{\Lambda})\boldsymbol{\Lambda}^{-1}$. Similarly

from $\mathbf{B} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T$ we have $\operatorname{diag}(\mathbf{U}^T\delta\mathbf{B}\mathbf{U}) = \frac{1}{2}\operatorname{diag}(\delta\boldsymbol{\Lambda})$. Continuing the derivation we have

$$\begin{aligned}
\frac{\partial f}{\partial \mathbf{B}} : \delta\mathbf{B} &= \frac{1}{2}\mathbf{U}^T\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\mathbf{U} : \operatorname{diag}(\delta\boldsymbol{\Lambda})\boldsymbol{\Lambda}^{-1} \\
&= \frac{1}{2}\mathbf{U}^T\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\mathbf{U} : \operatorname{diag}(\mathbf{U}^T\delta\mathbf{B}\mathbf{U})\boldsymbol{\Lambda}^{-1} \\
&= \frac{1}{2}\mathbf{U}^T\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\mathbf{U}\boldsymbol{\Lambda}^{-1} : \operatorname{diag}(\mathbf{U}^T\delta\mathbf{B}\mathbf{U}) \\
&= \frac{1}{2}\mathbf{U}^T\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\mathbf{U}\boldsymbol{\Lambda}^{-1} : \mathbf{U}^T\delta\mathbf{B}\mathbf{U} \\
&= \frac{1}{2}\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\mathbf{U}\boldsymbol{\Lambda}^{-1}\mathbf{U}^T : \delta\mathbf{B} \\
&= \frac{1}{2}\frac{\partial \hat{f}}{\partial \boldsymbol{\varepsilon}}\mathbf{B}^{-1} : \delta\mathbf{B}
\end{aligned}$$

Thus $\frac{\partial f}{\partial \mathbf{B}} = \frac{1}{2}\frac{\partial \hat{f}}{\partial \varepsilon}\mathbf{B}^{-1}$ which yields $\frac{\partial \hat{f}}{\partial \varepsilon} = 2\frac{\partial f}{\partial \mathbf{B}}\mathbf{B}$.

4.14.5 The relationship between Kirchhoff stress and Hencky strain

Claim: For any isotropic constitutive model ψ, $\boldsymbol{\tau} = \frac{\partial \psi}{\partial \varepsilon}$.

Proof: We have

$$\begin{aligned}
\mathbf{P} &= \frac{\partial \psi}{\partial \mathbf{F}} \\
\mathbf{P} &= \frac{\partial \tilde{\psi}}{\partial \mathbf{B}}\mathbf{F} + \mathbf{F}^T\frac{\partial \tilde{\psi}}{\partial \mathbf{B}} \\
\mathbf{P} &= 2\frac{\partial \tilde{\psi}}{\partial \mathbf{B}}\mathbf{F} \\
\mathbf{P}\mathbf{F}^T &= 2\frac{\partial \tilde{\psi}}{\partial \mathbf{B}}\mathbf{F}\mathbf{F}^T \\
\boldsymbol{\tau} &= 2\frac{\partial \tilde{\psi}}{\partial \mathbf{B}}\mathbf{B}.
\end{aligned}$$

By the Hencky strain derivative lemma applied to $\tilde{\psi}(\mathbf{B})$ we have $\boldsymbol{\tau} = \frac{\partial \psi}{\partial \varepsilon}$.

4.14.6 Plastic Dissipation is Nonnegative

Recall that we have previously defined $\mathbf{s} = \boldsymbol{\tau} - \frac{1}{d}\operatorname{tr}(\boldsymbol{\tau})$, and that $\mathbf{G} = \frac{\partial y}{\partial \boldsymbol{\tau}} - \frac{1}{d}\operatorname{tr}(\frac{\partial y}{\partial \boldsymbol{\tau}})\mathbf{I}$, i.e. it satisfies $\mathbf{G} = -\gamma \mathcal{L}_{\mathbf{v}} \mathbf{b}^E \mathbf{b}^{E^{-1}}$ (e.g. see Section (§4.12.2)). Therefore

$$\begin{aligned}
\dot{w}^P &= \boldsymbol{\tau} : \mathbf{l}^P \\
&= -\boldsymbol{\tau} : \frac{1}{2}\mathcal{L}_{\mathbf{v}} \mathbf{b}^E \mathbf{b}^{E^{-1}} \\
&= \gamma \boldsymbol{\tau} : \mathbf{G} \\
&= \frac{\gamma}{\|\mathbf{s}\|_F} \boldsymbol{\tau} : \mathbf{s} \\
&= \frac{\gamma}{\|\mathbf{s}\|_F} \left(\mathbf{s} + \frac{1}{d}\operatorname{tr}(\boldsymbol{\tau})\mathbf{I} \right) : \mathbf{s} \\
&= \gamma \|\mathbf{s}\|_F .
\end{aligned}$$

Thus all that remains to prove is that $\gamma \geq 0$. To do this we use the constraint that $\frac{\partial y}{\partial t} \leq 0$ when $y = 0$.

$$\begin{aligned}
\frac{\partial y}{\partial t} &= \frac{\partial y}{\partial \boldsymbol{\tau}} : \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{b}^E} : \dot{\mathbf{B}}^E \\
&= \frac{\partial y}{\partial \boldsymbol{\tau}} : \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{b}^E} : \left(\mathbf{l}\mathbf{b}^E + \mathbf{b}^E \mathbf{l}^T - 2\gamma \frac{\partial y}{\partial \boldsymbol{\tau}} \mathbf{b}^E \right) \\
&= \underbrace{\frac{\partial y}{\partial \boldsymbol{\tau}} : \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{b}^E} : \left(\mathbf{l}\mathbf{b}^E + \mathbf{b}^E \mathbf{l}^T \right)}_{\eta} - 2\gamma \underbrace{\frac{\partial y}{\partial \boldsymbol{\tau}} : \frac{\partial \boldsymbol{\tau}}{\partial \mathbf{b}^E} : \left(\frac{\partial y}{\partial \boldsymbol{\tau}} \mathbf{b}^E \right)}_{\nu} .
\end{aligned}$$

So

$$0 = \eta - 2\gamma\nu \Longrightarrow \gamma = \frac{\eta}{2\nu}$$

Note that η is what $\frac{\partial y}{\partial t}$ would be in the absence of plastic flow. Thus if $\eta \leq 0$ the material is deforming in such a way that the yield function is going down, and therefore is undergoing

elastic deformation which means $\gamma = 0$. Otherwise $\eta > 0$ and

$$\begin{aligned}\nu &= \frac{\partial y}{\partial \mathbf{b}^E} : \left(2 \frac{\mathbf{s}}{\|\mathbf{s}\|_F} \mathbf{b}^E \right) \\ &= 2 \frac{\partial y}{\partial \mathbf{b}^E} \mathbf{b}^E : \frac{\mathbf{s}}{\|\mathbf{s}\|_F}.\end{aligned}$$

Applying the Hencky strain derivative lemma to y we have

$$\begin{aligned}\nu &= \frac{\partial y}{\partial \boldsymbol{\varepsilon}^E} : \frac{\mathbf{s}}{\|\mathbf{s}\|_F} \\ &= \frac{\partial y}{\partial \boldsymbol{\tau}} : \frac{\partial \boldsymbol{\tau}}{\partial \boldsymbol{\varepsilon}^E} : \frac{\mathbf{s}}{\|\mathbf{s}\|_F} \\ &= \left(\frac{\mathbf{s}}{\|\mathbf{s}\|} + \tilde{\eta} \mathbf{I} \right) : \mathbb{C} : \frac{\mathbf{s}}{\|\mathbf{s}\|_F} \\ &= 2\mu \|\mathbf{s}\|_F.\end{aligned}$$

References

[AO11] I. Alduán and M. Otaduy. "SPH granular flow with friction and cohesion." In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 25–32, 2011. 9

[ATO09] Iván Alduán, A. Tena, and M. Otaduy. "Simulation of high-resolution granular media." In *Proc Cong Español Inf Graf*, 2009. 10

[BB08] C. Batty and R. Bridson. "Accurate Viscous Free Surfaces for Buckling, Coiling, and Rotating Liquids." In *Proc 2008 ACM/Eurographics Symp Comp Anim*, pp. 219–228, 2008. 8, 52

[BBS00] S. Bardenhagen, J. Brackbill, and D. Sulsky. "The material-point method for granular materials." *Comp Meth App Mech Eng*, **187**(3–4):529–541, 2000. 5, 10

[BFA02] Robert Bridson, Ronald Fedkiw, and John Anderson. "Robust treatment of collisions, contact and friction for cloth animation." In *ACM Trans. Graph. (ToG)*, volume 21, pp. 594–603. ACM, 2002. 1

[BH11] C. Batty and B. Houston. "A Simple Finite Volume Method for Adaptive Viscous Liquids." In *Proc 2011 ACM SIGGRAPH/Eurograp Symp Comp Anim*, pp. 111–118, 2011. 8, 52

[BIT09] M. Becker, M. Ihmsen, and M. Teschner. "Corotated SPH for Deformable Solids." In *Eurographics Conf. Nat. Phen.*, pp. 27–34, 2009. 8

[BMF03] R. Bridson, S. Marino, and R. Fedkiw. "Simulation of Clothing with Folds and Wrinkles." In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, SCA '03, pp. 28–36, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. 1

[BUA12] C. Batty, A. Uribe, B. Audoly, and E. Grinspun. "Discrete Viscous Sheets." **31**(4):113:1–113:7, 2012. 8, 52

[BW97] J. Bonet and R. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, 1997. 44, 45

[BW98] David Baraff and Andrew Witkin. "Large Steps in Cloth Simulation." In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pp. 43–54, New York, NY, USA, 1998. ACM. 1

[BW08] J. Bonet and R. Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge University Press, 2008. 58, 71, 72, 77, 95, 100, 101, 104

[BWH07] A. Bargteil, C. Wojtan, J. Hodgins, and G. Turk. "A Finite Element Method for Animating Large Viscoplastic Flow." *ACM Trans. Graph.*, **26**(3), 2007. 8

[BYM05] N. Bell, Y. Yu, and P. Mucha. "Particle-based simulation of granular materials." In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 77–86, 2005. 10

[CB02] S. Cummins and J. Brackbill. "An implicit Particle-in-Cell method for granular materials." *J Comp Phys*, **180**(2):506–548, 2002. 10

[CBL09] Y. Chang, K. Bao, Y. Liu, J. Zhu, and E. Wu. "A Particle-based Method for Viscoelastic Fluids Animation." In *ACM Symp. Virt. Real. Soft. Tech.*, pp. 111–117, 2009. 8

[CBZ12] Y. Chang, K. Bao, J. Zhu, and E. Wu. "A particle-based method for granular flow simulation." *Sci China Inf Sci*, **55**(5):1062–1072, 2012. 9

[CK05] Kwang-Jin Choi and Hyeong-Seok Ko. "Stable but responsive cloth." In *ACM SIGGRAPH 2005 Courses*, p. 1. ACM, 2005. 1

[CLH96] B. Chanclou, A. Luciani, and A. Habibi. "Physical models of loose soils dynamically marked by a moving object." In *Comp Anim*, pp. 27–35, 1996. 11

[CMH02] M. Carlson, P. Mucha, R. Van Horn, and G. Turk. "Melting and Flowing." In *ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, pp. 167–174, 2002. 8

[CPS10] Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. "A simple geometric model for elastic deformations." *ACM Trans. Graph. (TOG)*, **29**(4):38, 2010. 5, 25

[CW13] P. Chen and S. Wong. "Real-time auto stylized sand art drawing." In *CAD Comp Graph*, pp. 439–440, 2013. 11

[DG96] M. Desbrun and M. Gascuel. "Smoothed Particles: A New Paradigm for Animating Highly Deformable Bodies." In *Eurographics Workshop Comp. Anim. Sim.*, pp. 61–76, 1996. 8

[DP52] D. Drucker and W. Prager. "Soil mechanics and plasticity analysis or limit design." *Quart App Math*, **10**:157–165, 1952. 7, 58

[EEH00] Bernhard Eberhardt, Olaf Etzmuß, and Michael Hauth. *Implicit-explicit schemes for fast animation with particle systems*. Springer, 2000. 2

[EKS03] O. Etzmuss, M. Keckeisen, and W. Strasser. "A fast finite element solution for cloth modeling." In *Proc. Pac. Graph.*, pp. 244–251, 2003. 25

[GBB09] D. Gerszewski, H. Bhattacharya, and A. Bargteil. "A Point-based Method for Animating Elastoplastic Solids." In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 133–138, 2009. 8

[GBO04] T. Goktekin, A. Bargteil, and J. O'Brien. "A Method for Animating Viscoelastic Fluids." *ACM Trans. Graph.*, **23**(3):463–468, 2004. 8

[GHF07] Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. "Efficient simulation of inextensible cloth." In *ACM Trans. on Graph. (TOG)*, volume 26, p. 49. ACM, 2007. 3

[GS08] O. Gonzalez and A. Stuart. *A first course in continuum mechanics*. Cambridge University Press, 2008. 57, 95, 100

[GSO10] M Gonzalez, B Schmidt, and M Ortiz. "Force-stepping integrators in Lagrangian mechanics." *Intl. J. for Num. Meth. in Engng.*, **84**(12):1407–1450, 2010. 2

[GSS15] T. Gast, C. Schroeder, A. Stomakhin, C. Jiang, and J. Teran. "Optimization Integrator for Large Time Steps." *IEEE Trans Vis Comp Graph*, **21**(10):1103–1115, 2015. xii, 51, 79

[GZO10] Jorge Gascón, Javier S Zurdo, and Miguel A Otaduy. "Constraint-based simulation of adhesive contact." In *Proc. Symp. Comp. Anim.*, pp. 39–44, 2010. 28

[HE01] Michael Hauth and Olaf Etzmuss. "A high performance solver for the animation of deformable objects using advanced numerical methods." In *Comp. Graph. Forum*, volume 20, pp. 319–328, 2001. 1, 2

[HFL01] Gentaro Hirota, Susan Fisher, Chris Lee, H Fuchs, et al. "An implicit finite element method for elastic solids in contact." In *Comp. Anim., 2001.*, pp. 136–254. IEEE, 2001. 1, 2, 3

[HR97] P. Hiemenz and R. Rajagopalan. *Principles of Colloid and Surface Chemistry*. Marcel Dekker, 1997. 4

[ITF04] G. Irving, J. Teran, and R. Fedkiw. "Invertible finite elements for robust simulation of large deformation." In *Proc. Symp. Comp. Anim.*, pp. 131–140, 2004. 25

[IWT13] M. Ihmsen, A. Wahl, and M. Teschner. "A Lagrangian framework for simulating granular material with high detail." *Comp Graph*, **37**(7):800–808, 2013. 9

[Jia15] Chenfanfu Jiang. *The material point method for the physics-based simulation of solids and fluids*. PhD thesis, University of California, Los Angeles, 2015. 7, 10, 11, 82

[JNB96] H. Jaeger, S. Nagel, and R. Behringer. "Granular solids, liquids, and gases." *Rev Mod Phys*, **68**:1259–1273, 1996. 5

[JSS15] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. "The Affine Particle-In-Cell Method." *ACM Trans Graph*, **34**(4):51:1–51:10, 2015. 7, 10, 11, 49, 63, 65, 82

[KAG05] R. Keiser, B. Adams, D. Gasser, P. Bazzi, P. Dutré, and M. Gross. "A Unified Lagrangian Approach to Solid-fluid Animation." In *Eurographics/IEEE VGTC Conf. Point-Based Graph.*, pp. 125–133, 2005. 8

[Kan99] Couro Kane. *Variational integrators and the Newmark algorithm for conservative and dissipative mechanical systems*. PhD thesis, caltech, 1999. 1, 2

[KGP16a] G. Klár, T. Gast, A. Pradhana, C. Fu, C. Schroeder, C. Jiang, and J. Teran. "Drucker-Prager Elastoplasticity for Sand Animation: Supplementary Technical Document." *ACM Trans Graph*, 2016. xii

[KGP16b] Gergely Klar, Theodore Gast, Andre Pradhana, Chuyuan Fu, Craig Schroeder, Chenfanfu Jiang, and Joseph Teran. "Drucker-Prager Elastoplasticity for Sand Animation." *ACM Trans. Graph.*, **35**(4), July 2016. xii

[KMO99] C Kane, Jerrold E Marsden, and M Ortiz. "Symplectic-energy-momentum preserving variational integrators." *J. Math. Phys.*, **40**:3353, 1999. 2, 3

[KSJ08] Danny M Kaufman, Shinjiro Sueda, Doug L James, and Dinesh K Pai. "Staggered projections for frictional contact in multibody systems." In *ACM Trans. Graph. (TOG)*, volume 27, p. 164. ACM, 2008. 28, 33

[KYT06] Liliya Kharevych, Weiwei Yang, Yiying Tong, Eva Kanso, Jerrold E Marsden, Peter Schröder, and Mathieu Desbrun. "Geometric, variational integrators for computer animation." In *Proc. Symp. Comp. Anim.*, pp. 43–51, 2006. 2, 27

[Lar99] Ronald G Larson. *The Structure and Rheology of Complex Fluids*. Oxford University Press: New York, 1999. 4, 45

[LBO13] Tiantian Liu, Adam W Bargteil, James F O'Brien, and Ladislav Kavan. "Fast simulation of mass-spring systems." *ACM Trans. Graph. (TOG)*, **32**(6):214, 2013. 1, 2

[LD09] T. Lenaerts and P. Dutré. "Mixing Fluids and Granular Materials." *Comp Graph Forum*, **28**(2):213–218, 2009. 9

[LHM95] A. Luciani, A. Habibi, and E. Manzotti. "A multi-scale physical model of granular materials." In *Proc Graph Int*, pp. 136–146, 1995. 10

[LIG06] F. Losasso, G. Irving, E. Guendelman, and R. Fedkiw. "Melting and Burning Solids Into Liquids and Gases." *IEEE Trans. Vis. Comp. Graph.*, **12**:343–352, 2006. 8

[LM93] X. Li and J. Moshell. "Modeling soil: realtime dynamic models for soil slippage and manipulation." In *Proc SIGGRAPH*, pp. 361–368, 1993. 11

[LMO04] A Lew, JE Marsden, M Ortiz, and M West. "Variational time integrators." *Int J Numer Meth Eng*, **60**(1):153–212, 2004. 2

[MAM14] C. Mast, P. Arduino, P. Mackenzie-Helnwein, and R. Miller. "Simulating granular column collapse using the Material Point Method." *Acta Geotech*, **10**(1):101–116, 2014. 7, 10, 58, 77, 82

[Mas13] C. Mast. *Modeling landslide-induced flow interactions with structures using the Material Point Method*. PhD thesis, University of Washington, 2013. 7, 10, 58, 73, 82, 84, 92, 96

[MG04] M. Müller and M. Gross. "Interactive virtual materials." In *Proc. Graph. Intl.*, pp. 239–246, 2004. 25

[MHN15] H. Mazhar, T. Heyn, D. Negrut, and A. Tasora. "Using Nesterov's method to accelerate multibody dynamics with friction and contact." *ACM Trans Graph*, **34**(3):32:1–32:14, 2015. 10

[Mil96] V. Milenkovic. "Position-based physics: simulating the motion of many highly interacting spheres and polyhedra." In *Proc SIGGRAPH*, pp. 129–136, 1996. 10

[MKN04] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. "Point Based Animation of Elastic, Plastic and Melting Objects." In *ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, pp. 141–151, 2004. 8

[MMC14] M. Macklin, M. Müller, N. Chentanez, and T. Kim. "Unified particle physics for real-time applications." *ACM Trans Graph*, **33**(4):153:1–153:12, 2014. 11, 83

[MP89] G. Miller and A. Pearce. "Globular dynamics: a connected particle system for animating viscous fluids." *Comp Graph*, **13**(3):305–309, 1989. 10

[MR02] I. Morrison and S. Ross. *Colloidal Dispersions: Suspensions, Emulsions and Foams*. Wiley Interscience, 2002. 4

[MSW13] D. Michels, G. Sobottka, and A. Weber. "Exponential Integrators for Stiff Elastodynamic Problems." In *ACM Trans. Graph. (TOG)*. ACM, 2013. 2

[MTG11] Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. "Example-based elastic materials." In *ACM Trans. Graph. (TOG)*, volume 30, p. 72. ACM, 2011. 1, 2, 3

[Mus14] Ken Museth. "A Flexible Image Processing Approach to the Surfacing of Particle-Based Fluid Animation (Invited Talk)." In *Mathematical Progress in Expressive Image Synthesis I*, volume 4 of *Mathematics for Industry*, pp. 81–84. 2014. 50

[MZS11] A. McAdams, Y. Zhu, A. Selle, M. Empey, R. Tamstorf, J. Teran, and E. Sifakis. "Efficient elasticity for character skinning with contact and collisions." *ACM Trans. Graph.*, **30**:37:1–37:12, 2011. 5, 25

[NGL10] R. Narain, A. Golas, and M. Lin. "Free-flowing granular materials with two-way solid coupling." *ACM Trans Graph*, **29**(6):173:1–173:10, 2010. 9, 80, 83

[NKK12] D. Nkulikiyimfura, J. Kim, and H. Kim. "A real-time sand simulation using a GPU." In *Comp Tech Inf Man*, volume 1, pp. 495–498, 2012. 9

[NW06] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer series in operations research and financial engineering. Springer, 2006. 17, 19, 69

[ON03] K. Onoue and T. Nishita. "Virtual Sandbox." In *Proc Pac Conf Comp Graph App*, pp. 252–262, 2003. 11

[PF02] D Parks and D Forsyth. "Improved integration for cloth simulation." In *Proc. of Eurographics*, 2002. 2

[PGM06] M. Pla-Castells, I. Garcia-Fernandez, and R. Martinez. "Interactive terrain simulation and force distribution models in sand piles." In *Cellular Automata*, volume 4173 of *Lecture Notes Comp Sci*, pp. 392–401. 2006. 11

[PK96] R. Prudhomme and S. Kahn. *Foams: Theory, Measurements, and Applications*. Marcel Dekker, 1996. 4

[PKM02] A Pandolfi, C Kane, JE Marsden, and M Ortiz. "Time-discretized variational formulation of non-smooth frictional contact." *Intl. J. Num. Meth. Engng.*, **53**:1801–1829, 2002. 15

[PPL06] A. Paiva, F. Petronetto, T. Lewiner, and G. Tavares. "Particle-Based Non-Newtonian Fluid Animation for Melting Objects." In *Conf. Graph. Patt. Images*, pp. 78–85, 2006. 8

[PPL09] A. Paiva, F. Petronetto, T. Lewiner, and G. Tavares. "Particle-based Viscoplastic Fluid/Solid Simulation." *Comp. Aided Des.*, **41**(4):306–314, 2009. 8

[REN04] N. Rasmussen, D. Enright, D. Nguyen, S. Marino, N. Sumner, W. Geiger, S. Hoon, and R. Fedkiw. "Directable Photorealistic Liquids." In *ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, pp. 193–202, 2004. 8

[RGJ15] D. Ram, T. Gast, C. Jiang, C. Schroeder, A. Stomakhin, J. Teran, and P. Kavehpour. "A material point method for viscoelastic fluids, foams and sponges." In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 157–163, 2015. xii, 10

[Sch94] L. Schramm. *Foams: Fundamentals and Applications in the Petroleum Industry*. ACS, 1994. 4

[SCS94] Deborah Sulsky, Zhen Chen, and Howard L Schreyer. "A particle method for history-dependent materials." *Comp Meth in App Mech Eng*, **118**(1):179–196, 1994. 7

[SG09] Ari Stern and Eitan Grinspun. "Implicit-explicit variational integration of highly oscillatory problems." *Multiscale Modeling & Simulation*, **7**(4):1779–1794, 2009. 2

[SHS12] Alexey Stomakhin, Russell Howes, Craig Schroeder, and Joseph M Teran. "Energetically consistent invertible elasticity." In *Proc. Symp. Comp. Anim.*, pp. 25–32, 2012. 23, 25, 26, 86

[SKB08] Michael Steffen, Robert M Kirby, and Martin Berzins. "Analysis and reduction of quadrature errors in the material point method (MPM)." *Int J Numer Meth Eng*, **76**(6):922–948, 2008. 64

[SM93] JC Simo and G Meschke. "A new class of algorithms for classical plasticity extended to finite strains. Application to geomaterials." *Computational Mechanics*, **11**(4):253–278, 1993. 96

[SOH99] R. Sumner, J. O'Brien, and J. Hodgins. "Animating sand, mud and snow." *Comp Graph Forum*, **18**(1):17–26, 1999. 11

[SSC13] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. "A Material Point Method for Snow Simulation." *ACM Trans. Graph.*, **32**(4):102:1–102:10, July 2013. 3, 5, 7, 8, 9, 10, 34, 35, 36, 38, 40, 45, 47, 48, 49, 52, 84

[SSF13] Jonathan Su, Rahul Sheth, and Ronald Fedkiw. "Energy Conservation for the Simulation of Deformable Bodies." *Visualization and Computer Graphics, IEEE Transactions on*, **19**(2):189–200, 2013. 2

[SSJ14] A. Stomakhin, C. Schroeder, C. Jiang, L. Chai, J. Teran, and A. Selle. "Augmented MPM for phase-change and varied materials." *ACM Trans. Graph.*, **33**(4):138:1–138:11, 2014. 8, 10, 52

[SSP07] B. Solenthaler, J. Schläfli, and R. Pajarola. "A Unified Particle Model for Fluid-Solid Interactions." *Comp. Anim. Virt. Worlds*, **18**(1):69–82, 2007. 8

[ST08] R. Schmedding and M. Teschner. "Inversion handling for stable deformable modeling." *Vis. Comp.*, **24**:625–633, 2008. 25

[STW92] Juan C Simo, N Tarnow, and KK Wong. "Exact energy-momentum conserving algorithms and symplectic schemes for nonlinear dynamics." *Comp Meth in App Mech Eng*, **100**(1):63–116, 1992. 2

[TF88a] D. Terzopoulos and K. Fleischer. "Deformable models." *Vis Comp*, **4**(6):306–331, 1988. 7

[TF88b] D. Terzopoulos and K. Fleischer. "Modeling Inelastic Deformation: Viscolelasticity, Plasticity, Fracture." *SIGGRAPH Comp Graph*, **22**(4):269–278, 1988. 7

[TFS08] J. Teran, Lisa Fauci, and Michael Shelley. "Peristaltic pumping and irreversibility of a Stokesian viscoelastic fluid." *Phys Fl*, **20**(7), 2008. 45

[TSI05] J. Teran, E. Sifakis, G. Irving, and R. Fedkiw. "Robust quasistatic finite elements and flesh simulation." In *Proc. Symp. Comp. Anim.*, pp. 181–190, 2005. 25, 26

[VM01] Pascal Volino and Nadia Magnenat-Thalmann. "Comparing efficiency of integration methods for cloth simulation." In *Comp. graph. Intl. 2001 Proc.*, pp. 265–272. IEEE, 2001. 1, 2

[WT08] C. Wojtan and G. Turk. "Fast Viscoelastic Behavior with Thin Features." *ACM Trans. Graph.*, **27**(3):47:1–47:8, 2008. 8

[WTG09] C. Wojtan, N. Thürey, M. Gross, and G. Turk. "Deforming Meshes That Split and Merge." *ACM Trans. Graph.*, **28**(3):76:1–76:10, 2009. 8, 52

[YHK08] R. Yasuda, T. Harada, and Y. Kawaguchi. "Real-time simulation of granular materials using graphics hardware." In *Comp Graph Imag Vis*, pp. 28–31, 2008. 10

[Yos03] Naoto Yoshioka. "A sandpile experiment and its implications for self-organized criticality and characteristic earthquake." *Earth, planets and space*, **55**(6):283–289, 2003. 81

[YSB15] Y. Yue, B. Smith, C. Batty, C. Zheng, and E. Grinspun. "Continuum foam: a material point method for shear-dependent flows." *ACM Trans. Graph.*, **34**(5):160:1–160:20, 2015. 9, 10, 45, 51, 52

[ZB05] Y. Zhu and R. Bridson. "Animating sand as a fluid." *ACM Trans Graph*, **24**(3):965–972, 2005. 9, 80

[ZJ11] Changxi Zheng and Doug L James. "Toward high-quality modal contact sound." In *ACM Transactions on Graphics (TOG)*, volume 30, p. 38. ACM, 2011. 34

[ZST10] Y. Zhu, E. Sifakis, J. Teran, and A. Brandt. "An efficient and parallelizable multigrid framework for the simulation of elastic solids." *ACM Trans. Graph.*, **29**:16:1–16:18, 2010. 25