UNIVERSITY OF CALIFORNIA

Los Angeles

Multiphase Simulation Using Material Point Method

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

by

Andre Pradhana

2017

ABSTRACT OF THE DISSERTATION

Multiphase Simulation Using Material Point Method

by

Andre Pradhana
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2017
Professor Joseph M. Teran, Chair

We present a discussion on how one can simulate sand as a continuum using elastoplasticity. We showed the efficacy of Drucker-Prager plasticity model and St. Venant Kirchhoff with Hencky strain to model sand. We discretized the continuum equation using Material Point Method (MPM). We also present a multi-species model for the simulation of gravity driven landslides and debris flows with porous sand and water interactions. We use continuum mixture theory to describe individual phases where each species individually obeys conservation of mass and momentum and they are coupled through a momentum exchange term. Water is modeled as a weakly compressible fluid and sand is modeled with an elastoplastic law whose cohesion varies with water saturation. We use Material Point Method to discretize the governing equations. We use two grids, corresponding to water and sand phase. The momentum exchange term in the mixture theory is relatively stiff and we use semi-implicit time stepping to avoid associated small time steps. Our semi-implicit treatment is explicit in plasticity and preserves symmetry of force linearizations. We develop a novel regularization of the elastic part of the sand constitutive model that better mimics plasticity during the implicit solve to prevent numerical cohesion artifacts that would otherwise have occurred. Lastly, we develop an improved return mapping for sand plasticity that prevents volume gain artifacts in the traditional Drucker-Prager model.

Finally, we revisit the problem of redistancing, which is native to the level set paradigms. We used an interesting alternative view that utilizes the Hopf-Lax formulation of the solution to the eikonal equation, as proposed by [LDO17, DO16]. In this approach, the signed distance at an arbitrary point is obtained without the need of distance information from neighboring points. We extend the work of Lee et al. [LDO17] to redistance functions defined via interpolation over a regular grid.

The dissertation of Andre Pradhana is approved.

Jeffrey D. Eldredge

Christopher R. Anderson

Andrea L. Bertozzi

Joseph M. Teran, Committee Chair

University of California, Los Angeles

2017

To The Redeemer,

You make all things new.

Table of Contents

1 Introduction **1**

2 Dry Sand **2**

2.1 Previous work 2

2.2 Single Phase Continuum Theory 4

2.2.1 Single Phase Kinematics 4

2.2.2 The deformation gradient 6

2.2.3 Single Phase Balance Laws 8

2.3 Elastic material 10

2.3.1 Constitutive Model 11

2.4 Plasticity 12

2.4.1 The Drucker-Prager yield condition 12

2.4.2 Stress in the presence of plasticity 17

2.4.3 Unilateral hyperelasticity 17

2.5 Discretization 20

2.5.1 Notation 21

2.5.2 Deriving the MPM discretization 24

2.5.3 Notes on discrete energy and force 28

2.5.4 Plasticity in discrete setting 29

2.5.5 Notes on volume change in projection 31

2.6 Algorithm 34

2.6.1 Transfer to grid 35

2.6.2 Grid update 36

2.6.3 Transfer to particles 37

2.6.4 Update particle state 38

2.6.5 Plasticity, hardening 40

2.6.6 Hardening 40

2.6.7 Implicit velocity update 42

2.6.8 Semi implicit velocity update 42

2.6.9 Collisions 43

2.6.10 Friction 45

2.6.11 Initialization 46

2.7 Results 46

2.8 Discussion and Limitations 48

3 Wet Sand and Water 50

3.1 Previous Work 50

3.2 Multiphase Continuum 52

3.3 How to model water 55

3.3.1 KKT Approach 55

3.3.2 Weak compressibility 59

3.4 How to model wet sand 59

3.4.1 Momentum exchange 60

3.4.2 Cohesion and Saturation 61

3.5 Discretization 62

3.5.1 Transfer to grid 64

3.5.2 Update Grids Momenta 64

3.5.3 Update Particles 66

3.6 Implementation and Results 68

3.7 Limitations and future work 69

4 Redistancing 71

4.1 Introduction 71

4.2 Previous works 72

4.3 Our approach in continuous setting 75

4.4 The Hopf-Lax Formula 76

4.5 Discrete Setting 77

4.5.1 Secant method 78

4.5.2 Solving Hopf-Lax formula 79

4.6 Results and discussions 83

4.6.1 Computing in a narrow band 83

4.6.2 Computing geometric quantities 84

4.7 Results 85

4.7.1 Scaling 86

4.7.2 Level set advection 87

References 90

List of Figures

2.1 **Hourglass.** Sand falls through the narrow neck of an hourglass, accumulating at the bottom. 3

2.2 **Deformation gradient.** Relationship between deformation and $\boldsymbol{F}$. 6

2.3 **Multiplication decomposition.** Relationship between $\boldsymbol{F}$, $\boldsymbol{F}^E$, and $\boldsymbol{F}^P$. . . 7

2.4 **Plastic and elastic deformation.** How $\boldsymbol{F}$, $\boldsymbol{F}^E$, and $\boldsymbol{F}^P$ transforms elemental vector $d\boldsymbol{X}$, $d\tilde{\boldsymbol{x}}$, and $d\boldsymbol{x}$. 8

2.5 **Avalanching.** Sand is poured from a spout into a pile in a lab (left) and with our method (right). Our simulation is able to capture avalanching phenomenon. 10

2.6 **Coulomb friction illustration.** An illustration of normal and frictional force at a point $\boldsymbol{x}$ given an imaginary plane with normal $\boldsymbol{n}$. 14

2.7 **Multiplier region**. The contour plot of the multiplier in *strain space.* Region A is where the multiplier is equal to 1, region C is where the multiplier is 0, and region B is where the multiplier transitions from 1 to 0 in C^2 manner. The green cone denotes the Drucker-Prager cone in principal strain space. . . 18

2.8 **Unilateral extension**. We demonstrate a 2D sand column collapse with different values of c_0 coefficient in the unilateral modified energy density function. From top to bottom, $c_0 = 1.1, 2, 2.5, 3$. The left image is of an early frame and the right is of a later frame. Here parameters a and b from Equation (2.47) are $a = -0.5$, and $b = 0$. As c_0 increases, the envelopes in the unilateral extension become increasingly poor approximations of the Drucker-Prager cone and we see volume loss artifacts. 19

2.9 **B-splines interpolation.** Cubic (blue) and quadratic (red) splines used for computing interpolation weights. 23

2.10 **Drucker-Prager region.** The yield surface for Drucker-Prager is shown in principal stretch space. 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. 30

2.11 **Sand column collapse.** A column of sand collapses into a pile. 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. . 31

2.12 **Pouring sand**. The top row depicts the result of an explicit simulation with a coarser grid size ($dx = 10^{-2}$), while the bottom row corresponds to a finer grid size ($dx = 10^{-3}$). The first and third column correspond to a projection step without volume correction, while the second and fourth column uses our volume correction algorithm. 32

2.13 **2D hourglass.** An hourglass is flipped two times. The left most hourglass shaded grey depicts the initial state of sand at the beginning of the simulation. Each set-of-three figures depicts the state of the sand after going through the neck the first time, and after flipping the hourglass for the first and second time. The first two sets and the last two sets have grid $dx = 7.4 \times 10^{-3}$ and $dx = 3.7 \times 10^{-3}$ respectively. The first and third set are run without any volume correction fix, while the second and last set are run with our volume fix. . . . 33

2.14 **Overview of MPM stages.** Breakdowns of an MPM algorithm. 34

2.15 **Raking.** A rake is dragged around a rock, producing a circular pattern in the sand. 35

2.16 **Effects of Young's modulus.** This simulation shows the effects of Young's modulus on the behavior of a simulation. 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. 37

2.17 **Avalanching.** Sand is poured from a spout into a pile in a lab (left) and with our method (right). Our simulation is able to capture avalanching phenomenon. 38

2.18 **Coupling with stiff material.** A solid ball drops into a sandbox, spraying sand in all directions. 39

2.19 **Hardening.** 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. 40

2.20 **Coupling with deformable object.** A sand castle is hit with a deformable ball while falling. The sand and ball are fully coupled in the simulation. 41

2.21 **Effects of friction angle on piling.** Varying the friction angle changes the shape of a pile of sand. A larger angle produces a taller sand pile with steeper sides. 43

2.22 **Drawing in sand.** A stick is dragged through a bed of sand, tracing out a butterfly shape in the sand. 44

2.23 **2D sand pile.** From top to bottom row: initial, middle, and final configuration of a 2D sand pile. The left column is an explicit simulation with maximum $\Delta t = 10^{-4}$. The two columns in the middle are the results of semi-implicit simulation with the regular constitutive model and max $\Delta t = 10^{-2}$ and 10^{-3} respectively. Note that both suffer from artificial cohesion, although it diminishes with smaller Δt. On the right is semi-implicit time stepping with our unilateral energy density function with maximum $\Delta t = 10^{-2}$. Note that it does not suffer from artificial cohesion. 47

2.24 **Fully implicit unilateral.** Comparison of increasingly implicit time stepping schemes for 2D sand column collapse. The simulations on the left and at the center are run with the semi-implicit scheme with the regular and unilateral energy density functions respectively. The right-most figure shows a fully implicit scheme with our modified energy density function. Note that the unilateral density function with semi-implicit time stepping yields a good approximation of the fully implicit result, while the regular energy density suffers from artificial cohesion. 48

3.1 **Dam breach**. Water pours in from a reservoir and slowly erodes a dam. As water seeps into the sand, its cohesivity decreases. When it eventually breaks, the landslide creates interesting dynamics in the debris flow. 50

3.2 **Dropping sand on wedges**. A box of sand with various values of cohesion is dropped onto a wedge. The top left simulation is done using an explicit time-stepping scheme, while all the others are done using a semi-implicit time-stepping scheme. 60

3.3 **MPM algorithm with two grids**. We use separate water and sand grids. The blue and red dots denote water and sand particles respectively. In the overlapping region, we compute the momentum exchange term between the two species. We also compute the force based on individual constitutive model. 62

3.4 **2D dam breach.** A 2D dam breach simulation showing the effect of reversible term from Bandara and Soga [BS15]. The figure depicts the beginning, middle, and the end of a dam breach scenario. At the top is a simulation with no reversible term (as in [MAS10]) and at the bottom is a simulation with the reversible term. 68

3.5 **When the levee breaks**. Here we demonstrate the effects of our parameters on levee wall integrity. A simulation with lower drag momentum exchange coefficient and lower cohesion will fail more easily. 69

4.1 **Local minima illustration.** The two vertical lines are the boundary of minimization. Grid node $x_i = 1.8$ is in the middle of the region, and also the starting guess for projected gradient descent. The sequence of points leading off to the right represent the subsequent steps of gradient descent. These points converge to the incorrect argmin $x = 2.5$. The correct solution is at $x = 0.4$. In order to converge to this point, the initial guess would have to be less than 1.25 81

4.2 **Random initial guess in 1D.** The figure illustrates representative random initial guesses used in solving for $\phi(\boldsymbol{x}_i, t^k)$. In addition we use an initial guess equal to the minimizer computed in the previous secant iteration shown in magenta. 82

4.3 **Different choice of random guesses and gradient discent iteration.** The plots above are the points $(\phi(\boldsymbol{x}_i, t^k), t^k)$ found when running our algorithm with different choices of random guess and gradient descent iterations on circle initial data. The left most plot was run with 100 random guesses, and 1 gradient descent iteration. The middle plot was run with 1 random guess, and 100 gradient descent iterations. The right plot was run with 1 random guess and 5 gradient descent iterations. Note that in all cases, the correct root was found. 83

4.4 **Random initial guesses in 2D.** In this image, the red dot in the center is $\boldsymbol{x}_i$, the solid red line represents the ball of radius t_k and the dotted line represents the ball of radius t_{k-1}. The magenta point was the approximate argmin $\boldsymbol{y}_{k-1}$ of ϕ^0 over the ball of radius t_{k-1}. Since it is unlikely for the minimizer to be inside of t_{k-1} we use coarse (random) grid initial guesses in the interior. However, since it is possible that expanding t will move the minimizer to a different location we take a large number of initial guesses along the boundary of t_k 84

4.5 **Narrow band and coarse grid.** A coarse grid 8 times smaller than the fine grid was solved for initially. Using those values the fine grid was only solved on cells where the distance to the boundary could be less than 0.1 represented as the solid areas of the left image. In the right image those coarse areas are defined from bilinear interpolation. This coarse/banding approach provided approximately a 2.5 times increase in performance. 85

4.6 **Scaled circle.** The initial data is $\phi^0 = e^x(0.125 - (0.5-x)^2 + (0.5-y)^2)$. . . . 86

4.7 **Square**. The initial data is $\phi^0 = \min\{0.25 - |x-0.5|, .25 - |y-0.5|\}$. 87

4.8 **Union of circles.** The initial data is $\phi^0 = \max\{0.25 - \|(0.3, 0.5) - (x,y)\|_2, 0.25 - \|(0.7, 0.5) - (x,y)\|_2\}$. 87

4.9 **Many local minima.** the objective function is $\phi^0 = \sin(4\pi x)\sin(4\pi y) - 0.01$. 88

4.10 **Vortex test.** Vortex advection test at $t = 0, 1, 2, 3, 4, 5$. 89

List of Tables

3.1 **Table of notation for multiphase simulation.** 63

4.1 **Parameters and timings.** Table of parameters for examples and timings. . 83

4.2 **Error.** Average ℓ_2 error with different grid resolution. 88

Acknowledgments

Thank You Lord Jesus, for this life, for meeting me in 2011 when it was the bleakest moment, thank You for inviting me to give mathematics another chance and for this graduate school opportunity, thank You for sustaining me day by day and for delivering me through grad school. I love You.

Thank you to my advisor Prof. Joseph Teran who has been so patient with me days in and days out, for being such a great researcher and mentor, for your sacrifice of going to the lab at night, and for our conversations on Man United. Thank you Prof. Chris Anderson for showing me the joy of doing applied mathematics and being the best math teacher I've ever had. Thank you to Prof. Bertozzi for all of your help during my graduate school career and for being in my defense even when you are still recovering from surgery. Thank you Prof. Eldredge for your enthusiasm on science and being a role model on being a scientist and mentor (I am always reminded of Caltech when I meet you) .

Greg: professionally, you introduced me to computer graphics and rendering, taught me `C++`, programming, software architecture, and much more. You and Zsófi showed me the joy of having a family. Thank you for all the random walks and the many conversations we had. Thank you so much for blessing me in so many ways.

Thank you to Craig Schroeder, Chenfanfu Jiang, and Ted Ghast for helping me a lot during the various stages of many projects and for showing me how research can be done. Thank you especially to Fanfu for offering me a job after graduation.

Thank you Ken for your mentorship and a chance to do a summer internship at DreamWorks. Thank you for showing me the joy of working in computer graphics. Thank you Malena Español for your patient and mentorship back at Caltech. Thank you Jared Hersh for being such a great TA, for showing me the cuteness of ODEs, and for convincing me to do mathematics.

Thank you to Joel Wretborn and Max Budninskiy for all of our conversations and for all of the fun that we had at DreamWorks. Thank you to Ibrahim for all your help with getting JIXIE to compile at DreamWorks. Thank you to Jeff Budsberg and Stephen Wood for showing me how to be passionate in doing your work and for all of the fun.

Thank you to Charlie and Nadine, and Daniel and Xiao. Grad school can be tough, but I've won a few friends there. That by itself makes it worth it.

Thank you Michael for our collaboration on redistancing. Thank you for the people in the lab: Qi for all of your help through the wet-sand paper and your always-positive attitude, Kelly for your help on unilateral, and Stephanie for always willing to substitute for `PIC10A`. Thank you Tomer for our friendship and feeding me so many times in the lab, Sharath for all of your encouragements, Masaki for all the fun that you brought to the lab, and Tao for your constant presence in the lab.

Thank you Cherrie for being such a close friend even long after Caltech. Thank you Roy Qi for our friendship and all your help during the Santa Barbara years and beyond. Jacob Houpis for being such a dear friend and for all the jokes.

Thank you Peter for the many times we prayed together and for taking care of all of our apartment needs. Thank you Tom for your presence in our apartment. Thank you Jimmy for the many times we hung out with each other and your prayers and encouragements. Thank you Daniel for our adventures to K-Town and coffee shops together, I've always enjoyed our conversations (every single one of them). Thank you David for being so positive, supportive, and always pointing me towards Jesus. Thank you Soaps for always being so encouraging and gentle. Thank you Jooeun and Kelly for showing me what a family life looks like and for being a United fan.

Thank you John for all your prayers, phone calls, encouragements, and the many dinners we had together. Thank you Joel for being such a close friend and a light in my life and being there with prayers and encouragements when I am weak.

I can't say enough to thank Dongyoon and Elaine. There is a friend that sticks closer than a brother. Thank you for loving me in so many ways, for cooking me lunch boxes for weeks leading up to my defense, for opening up your home time and time again, for our study time, for all your encouragements, and prayers.

Thank you P. Abraham and Sarah for your prayers and blessings every time we met when I was at Caltech, Cambridge, and whenever we met in the LA area again.

Thank you to P. Ray and Jackie and your family. What can I say? Thank you for your love, prayers, blessings, encouragements, and for keeping me accountable for so many years. I can't ask for a better shepherd, mentor, and friend.

Thank you to Mom and Dad and Mba Thia. Thank you so much for all of your unconditional love, prayers, and for just being there. Mom and Dad, thank you for encouraging me to pursue education, even to this point. Thank you for being the most amazing parents a child could ask for and for all the amazing memories in Indonesia, Europe, and the U.S. Mba, I always cherish our time in England together, and I love you. It's really a privilege to be part of this family, and I've won a lottery without buying a ticket.

There are many more people who have blessed me in so many ways over the past five years. Unfortunately this list must end somewhere. I just want to say that I am thankful for every single one of you.

I hope we'll all talk about it again in the by and by.

Vita

2011	B.S. (Applied and Computational Mathematics), California Institute of Technology.
2012	Master of Advanced Study (DAMTP), University of Cambridge.
2016	Summer Intern, DreamWorks Animation Studios

Publications

G. Klár, 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

A. Pradhana Tampubolon, T. F. Gast, G. Klár, C. Fu, J. Teran, C. Jiang, and K. Museth. "Multispecies Simulation of Porous Sand and Water Mixtures." ACM Trans. Graph. (SIGGRAPH), 2017

M. Royston, A. Pradhana, B. Lee, Y. T. Chow, W. Yin, J. Teran, and S. Osher. "Parallel redistancing using the Hopf-Lax formula." Submitted, 2017

CHAPTER 1

Introduction

I remember the day when my advisor called me into his office, and he told me, "We have to do wet sand!"

Later on, during my Advancement-To-Candidacy talk, I showed a YouTube video about a Lego dam breach in front of my committee members. My advisor claimed that we want to do a simulation like that. This thesis tells that story.

Chapters 2 and 3 are slight modifications of [KGP16] and [PGK17]. They are supposed to tell a story of how one can simulate a dam breach using Material Point Method (MPM). Chapter 2 tells a story of how to simulate dry sand as an elastoplastic material using MPM. One can view it as a study in plasticity as a continuum model of friction. Chapter 3 tells a story of one way we can simulate water and wet sand using MPM, and how they can interact.

Chapter 4 is a slight modification of [RPL17]. This chapter is an oddity on its own, since it is a revisiting of an old problem in level set, which is how to do redistancing. However, with a modest tweak, one can show that the method looks promising as an alternative to do a return-mapping algorithm in computational plasticity. Finally, the last chapter is a study on parallel computation using GPU, where the future of scientific computing might lie.

CHAPTER 2

Dry Sand

Before we attempt to simulate wet sand, we need to figure out how to simulate dry sand. If we model sand as a continuum, to a great extent, sand can be thought of as an elastoplastic material. To a first order approximation one can think that up to a certain point, the deformation of sand is governed by elasticity. Beyond this, plasticity imposes a certain restriction on the grain of sand applies friction and slide against one another. This chapter tells the story of simulating dry sand.

2.1 Previous work

We build on the work of Mast et al. [Mas13, MAM14a] 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 model 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 [JSS15] and show that this allows for more stable behavior, particularly with simulations that have higher numbers of particle per cell.

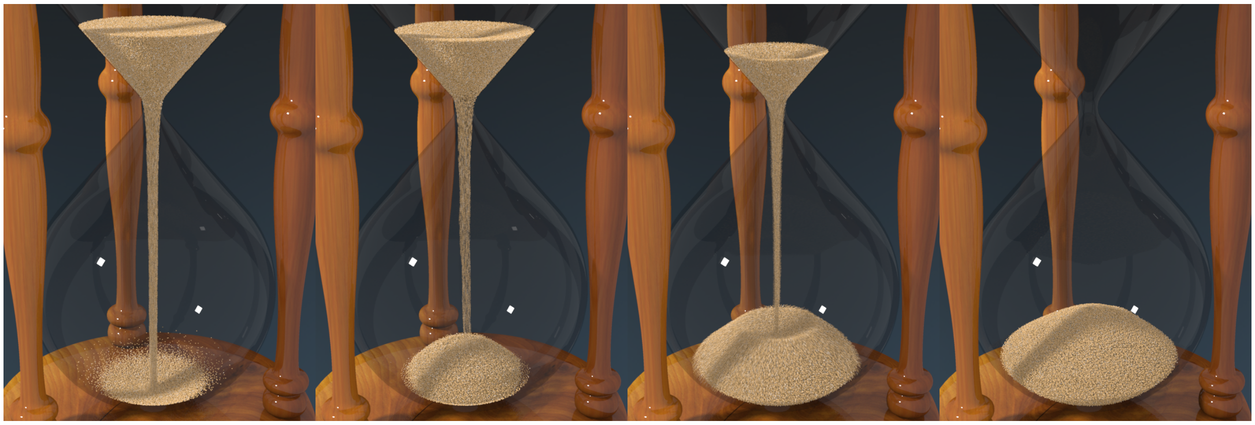

Figure 2.1: ***Hourglass.*** *Sand falls through the narrow neck of an hourglass, accumulating at the bottom.*

Continuum approaches have been used in a number of graphics methods for granular materials. Zhu and Bridson [ZB05] animate sand as a continuum with a modified Particle-In-Cell fluid solver. Narain et al. [NGL10] improve on the method of Zhu and Bridson by removing cohesion artifacts associated with incompressibility. 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 [LD09] use an SPH version to couple water with porous granular materials. Alduán and Otaduy [AO11] generalize the unilateral incompressibility developed by Narain et al. to SPH. Imhsen et al. [IWT13] show how to improve the convergence of the method of Alduán and Otaduy [AO11] and also detail refinement of base simulations to upscale to millions of grains. Chang et al. [CBZ12] use a modified Hooke's law to handle friction between grains.

Many methods are developed by modeling interactions between individual grains or particle idealizations of grains, rather than from a continuum. Miller and Pearce [MP89] simulate interactions between particles to model sand, solid and viscous behaviors. Luciani et al. [LHM95] use a similar approach. Bell et al. [BYM05] got very impressive results by simulating many sand grains as spherical rigid bodies with friction. Milenkovic [Mil96] also simulated individual grains to solve for piles of rigid materials via energy minimization/optimization. Mazhar et al. [MHN15] use Nestov's method to simulate millions of individual grains. Yasuda et al. [YHK08] use the GPU to get real-time results with rigid grains. Alduan et al. [ATO09] use an adaptive resolution version of the method by Bell et al. [BYM05] to improve performance. Macklin et al. [MMC14] show that the extremely efficient position based dynamics methods can be applied by casting granular interactions as hard constraints in.

2.2 Single Phase Continuum Theory

To develop the theory for two phases, we first begin with the theory of kinematics and continuum of one phase.

2.2.1 Single Phase Kinematics

There are two standard ways of working with the kinematics of a motion: the Lagrangian and Eulerian description. In the Lagrangian viewpoint, one keeps track of the motion of each point from its initial configuration over time. The motion of a material with respect to its initial configuration is described by a flow map $\boldsymbol{\varphi}(\cdot,t):\Omega^0\rightarrow\Omega^t$, which is assumed to be a smooth bijection (diffeomorphism). It maps a point $\boldsymbol{X}$ in a reference configuration to a point $\boldsymbol{x}$ in the current configuration, i.e.

$$\boldsymbol{x}(t)=\boldsymbol{\varphi}(\boldsymbol{X},t),\quad\text{or}\quad\boldsymbol{X}(t)=\boldsymbol{\varphi}^{-1}(\boldsymbol{x},t).\tag{2.1}$$

This maps naturally defines the notion of velocity and acceleration:

$$\boldsymbol{V}(\boldsymbol{X},t):=\frac{\partial\boldsymbol{\varphi}}{\partial t}(\boldsymbol{X},t),\quad\boldsymbol{A}:=\frac{\partial\boldsymbol{V}}{\partial t}(\boldsymbol{X},t).\tag{2.2}$$

The function $\boldsymbol{V}(\cdot,t)$ and $\boldsymbol{A}(\cdot,t)$ are parametrized by the reference configuration, and are considered to be Lagrangian quantities. We define their Eulerian counterparts, which is defined in terms of world-space configuration

$$\boldsymbol{v}(\boldsymbol{x},t) \coloneqq \boldsymbol{V}(\boldsymbol{\varphi}^{-1}(\boldsymbol{x},t),t), \quad \text{i.e.} \quad \boldsymbol{v}(\boldsymbol{\varphi}(\boldsymbol{X},t),t) \coloneqq \boldsymbol{V}(\boldsymbol{X},t). \tag{2.3}$$

Acceleration can also be similarly defined as

$$\boldsymbol{a}(\boldsymbol{x},t) \coloneqq \boldsymbol{A}(\boldsymbol{\varphi}^{-1}(\boldsymbol{x},t),t), \quad \text{i.e.} \quad \boldsymbol{a}(\boldsymbol{\varphi}(\boldsymbol{X},t),t) \coloneqq \boldsymbol{A}(\boldsymbol{X},t). \tag{2.4}$$

These can be thought of as the push-forward of their Lagrangian counterparts.

If we consider a total derivative of the push-forward velocity $\boldsymbol{v}(\boldsymbol{x},t)$ with respect to time, we get

$$\begin{aligned}
\frac{D}{Dt}\boldsymbol{v}(\boldsymbol{x},t) &= \frac{D}{Dt}\boldsymbol{v}(\boldsymbol{\varphi}(\boldsymbol{X},t),t) && (2.5)\\
&= \nabla^{\boldsymbol{x}}\boldsymbol{v}(\boldsymbol{\varphi}(\boldsymbol{X},t),t)\boldsymbol{V}(\boldsymbol{X},t) + \frac{\partial \boldsymbol{v}}{\partial t}(\boldsymbol{\varphi}(\boldsymbol{X},t),t) && (2.6)\\
&= \nabla^{\boldsymbol{x}}\boldsymbol{v}(\boldsymbol{\varphi}(\boldsymbol{X},t),t)\boldsymbol{v}(\boldsymbol{\varphi}(\boldsymbol{X},t),t) + \frac{\partial \boldsymbol{v}}{\partial t}(\boldsymbol{\varphi}(\boldsymbol{X},t),t) && (2.7)\\
&= \nabla^{\boldsymbol{x}}\boldsymbol{v}(\boldsymbol{x},t)\boldsymbol{v}(\boldsymbol{x},t) + \frac{\partial \boldsymbol{v}}{\partial t}(\boldsymbol{x},t). && (2.8)
\end{aligned}$$

It is understood that we have used $\boldsymbol{x} = \boldsymbol{\varphi}(\boldsymbol{X},t)$. We call this type of derivative the material derivative. For a generic function $h(\boldsymbol{x},t) : \Omega^t \times (0,T) \to \mathbb{R}$, the material derivative of h is given by

$$\frac{Dh}{Dt}(\boldsymbol{x},t) = \frac{\partial h}{\partial t}(\boldsymbol{x},t) + \nabla^{\boldsymbol{x}}h(\boldsymbol{x},t)\cdot\boldsymbol{v}(\boldsymbol{x},t), \tag{2.9}$$

and for $\boldsymbol{h}(\boldsymbol{x},t) : \Omega^t \times (0,T) \to \mathbb{R}^d$, the material derivative of $\boldsymbol{h}$ is

$$\frac{D\boldsymbol{h}}{Dt}(\boldsymbol{x},t) = \frac{\partial \boldsymbol{h}}{\partial t}(\boldsymbol{x},t) + \nabla^{\boldsymbol{x}}\,\boldsymbol{h}(\boldsymbol{x},t)\boldsymbol{v}(\boldsymbol{x},t). \tag{2.10}$$

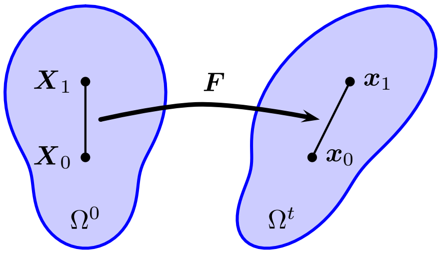

Figure 2.2: ***Deformation gradient.*** *Relationship between deformation and* $\boldsymbol{F}$*.*

2.2.2 The deformation gradient

We define the deformation gradient tensor as

$$\boldsymbol{F}(\boldsymbol{X},t) \coloneqq \frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{X}}(\boldsymbol{X},t) = \nabla^{\boldsymbol{X}} \boldsymbol{\varphi}(\boldsymbol{X},t). \tag{2.11}$$

It represents the infinitesimal local deformation of a body. Given the difference between two points in the original configuration of a body: $\boldsymbol{X}_1 - \boldsymbol{X}_0$, and its corresponding image in the current configuration $\boldsymbol{x}_1 - \boldsymbol{x}_0$, these differences are related in this way: $\boldsymbol{x}_1 - \boldsymbol{x}_0 = \boldsymbol{F}(\boldsymbol{X}_1 - \boldsymbol{X}_0)$. This is illustrated by Figure 2.2.

The determinant of the deformation gradient is denoted by $J(\boldsymbol{X},t)$.

2.2.2.1 Multiplicative decomposition

Often, it is beneficial to factor the deformation gradient into elastic and plastic parts as $\boldsymbol{F} = \boldsymbol{F}^E \boldsymbol{F}^P$. The *full* 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, $\boldsymbol{F}^P$, 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 2.3). The twisting and bending involved in this operation is stored in $\boldsymbol{F}^P$. If the spring is compressed slightly, the spring will feel strain (deformation). This is elastic deformation, which is stored in $\boldsymbol{F}^E$. The spring remembers this deformation. In response, the material exerts stress to try to restore itself to its coiled shape. In this way,

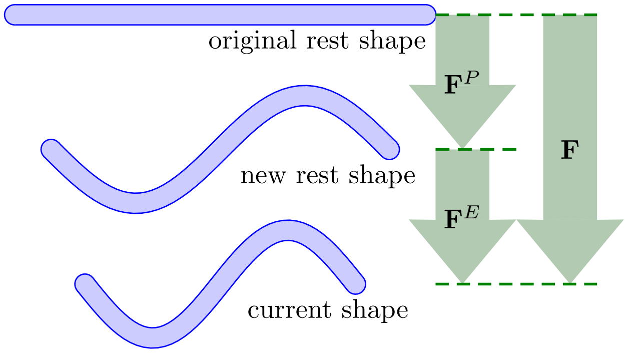

Figure 2.3: ***Multiplication decomposition.*** *Relationship between $\boldsymbol{F}$, $\boldsymbol{F}^E$, and $\boldsymbol{F}^P$.*

we see that only $\boldsymbol{F}^E$ should be used to compute stress. The full history of the metal rod consists of being bent into a spring shape ($\boldsymbol{F}^P$) and then being compressed ($\boldsymbol{F}^E$). For sand, $\boldsymbol{F}^E$ represents the remembered compression and shearing, while $\boldsymbol{F}^P$ represents the forgotten sliding and separation.

As in [BW08], we explain the mathematical consequence of this decomposition as follows. Let $\boldsymbol{X}$ be a point in the reference configuration, and let $d\boldsymbol{X}$ be an elemental vector in a local neighborhood of $\boldsymbol{X}$. The point $\boldsymbol{X}$ will be mapped to $\boldsymbol{x}$ under $\boldsymbol{\varphi}$, and the elemental vector $d\boldsymbol{x}$ will be related to $d\boldsymbol{X}$ as

$$d\boldsymbol{x} = \boldsymbol{F} d\boldsymbol{X}. \tag{2.12}$$

However, the elastic rest state of $\boldsymbol{x}$ is defined by some other configuration, and we postulate that there exists $\boldsymbol{F}^E$ and $\boldsymbol{F}^P$ such that

$$d\tilde{\boldsymbol{x}} = \boldsymbol{F}^P d\boldsymbol{X} \quad \text{and} \quad d\boldsymbol{x} = \boldsymbol{F}^E d\tilde{\boldsymbol{x}}, \tag{2.13}$$

i.e. $\boldsymbol{F}^P$ carries $d\boldsymbol{X}$ to a new state $d\tilde{\boldsymbol{x}}$, which defines a new elastic rest state, and the final elemental vector $d\boldsymbol{x}$ is given by $\boldsymbol{F}^E d\tilde{\boldsymbol{x}}$. This is illustrated in Figure 2.4.

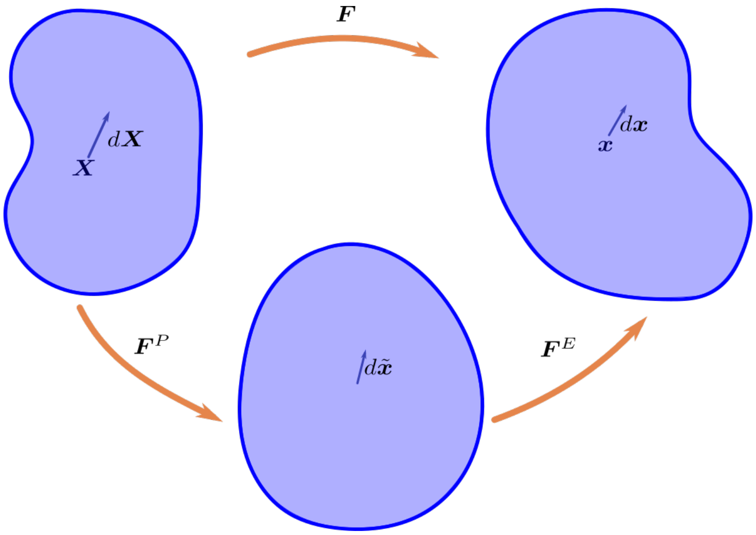

Figure 2.4: ***Plastic and elastic deformation.*** *How $\boldsymbol{F}$, $\boldsymbol{F}^E$, and $\boldsymbol{F}^P$ transforms elemental vector $d\boldsymbol{X}$, $d\tilde{\boldsymbol{x}}$, and $d\boldsymbol{x}$.*

2.2.2.2 The evolution of the deformation gradient

We note the following useful equation dictating the evolution of the deformation gradient, namely

$$\frac{\partial \boldsymbol{F}}{\partial t}(\boldsymbol{X},t) = \nabla^{\boldsymbol{x}} \boldsymbol{v}(\boldsymbol{x},t)\Big|_{\boldsymbol{x}=\boldsymbol{\varphi}(\boldsymbol{X},t)} \boldsymbol{F}(\boldsymbol{X},t), \tag{2.14}$$

while the evolution of its determinant is governed by

$$\frac{\partial J}{\partial t}(\boldsymbol{X},t) = J(\boldsymbol{X},t)\, \nabla^{\boldsymbol{x}} \cdot \boldsymbol{v}(\boldsymbol{x},t)\Big|_{\boldsymbol{x}=\boldsymbol{\varphi}(\boldsymbol{X},t)}. \tag{2.15}$$

2.2.3 Single Phase Balance Laws

Mass-balance. We assume the existence of a mass density field $R(\boldsymbol{X},t)$ for our continuum, and define $\rho(\boldsymbol{x},t)$ as the push-forward of this Lagrangian density field.

$$\rho(\boldsymbol{x},t) := R(\boldsymbol{\varphi}^{-1}(\boldsymbol{x},t),t), \quad \text{i.e.} \quad \rho(\boldsymbol{\varphi}(\boldsymbol{X},t),t) := R(\boldsymbol{X},t). \tag{2.16}$$

Conservation of mass is the statement that the integral of mass density over all control volume in the reference configuration and the world space should be the same, i.e. if $\tilde{\Omega}^0$ is any subset of Ω^0 and $\tilde{\Omega}^t$ is its image under $\boldsymbol{\varphi}$, then

$$\int_{\tilde{\Omega}^0} R(\boldsymbol{X},0)\,\mathrm{d}\boldsymbol{X} = \int_{\tilde{\Omega}^t} \rho(\boldsymbol{x},t)\,\mathrm{d}\boldsymbol{x} = \int_{\tilde{\Omega}^0} J(\boldsymbol{X},t)\,R(\boldsymbol{X},t)\,\mathrm{d}\boldsymbol{X}, \tag{2.17}$$

with the first equality *is* a statement of mass conservation and the second comes from a change of variable. But this is true for any control volume $\tilde{\Omega}^0$, so

$$R(\boldsymbol{X},0) = R(\boldsymbol{X},t)J(\boldsymbol{X},t). \tag{2.18}$$

The above is the Lagrangian formulation of mass-balance. To get the Eulerian counterparts, we take the derivative of Equation (2.18) with respect to time:

$$\begin{aligned}
0 &= \frac{\partial}{\partial t}(R(\boldsymbol{X},t)J(\boldsymbol{X},t)) \\
&= \frac{\partial R}{\partial t}(\boldsymbol{X},t)J(\boldsymbol{X},t) + \frac{\partial J}{\partial t}(\boldsymbol{X},t)R(\boldsymbol{X},t) \\
&= \frac{D\rho}{Dt}(\boldsymbol{x},t)\Big|_{\boldsymbol{x}=\boldsymbol{\varphi}(\boldsymbol{X},t)} J(\boldsymbol{X},t) + \frac{\partial J}{\partial t}(\boldsymbol{X},t)\rho(\boldsymbol{\varphi}(\boldsymbol{X},t),t) \\
&= \frac{D\rho}{Dt}(\boldsymbol{x},t)\Big|_{\boldsymbol{x}=\boldsymbol{\varphi}(\boldsymbol{X},t)} J(\boldsymbol{X},t) + J(\boldsymbol{X},t)\,\nabla^{\boldsymbol{x}}\cdot\boldsymbol{v}(\boldsymbol{x},t)\Big|_{\boldsymbol{x}=\boldsymbol{\varphi}(\boldsymbol{X},t)}\rho(\boldsymbol{\varphi}(\boldsymbol{X},t),t) \\
&= \left[\frac{D\rho}{Dt}(\boldsymbol{x},t)J(\boldsymbol{X},t) + J(\boldsymbol{X},t)\,\nabla^{\boldsymbol{x}}\cdot\boldsymbol{v}(\boldsymbol{x},t)\rho(\boldsymbol{x},t)\right]\Big|_{\boldsymbol{x}=\boldsymbol{\varphi}(\boldsymbol{X},t)},
\end{aligned}$$

and dividing both sides by $J(\boldsymbol{X},t)$, we arrive at

$$\frac{D\rho}{Dt}(\boldsymbol{x},t) + \nabla^{\boldsymbol{x}}\cdot\boldsymbol{v}(\boldsymbol{x},t)\rho(\boldsymbol{x},t) = 0.$$

Momentum-balance. For the model of deformable bodies that we are interested in, we assume the existence of a stress field $\boldsymbol{P}(\boldsymbol{F}(\boldsymbol{X},t))$ which capture the internal traction in the body. We call this stress tensor the *first Piola-Kirchhoff* tensor. Hence, the momentum

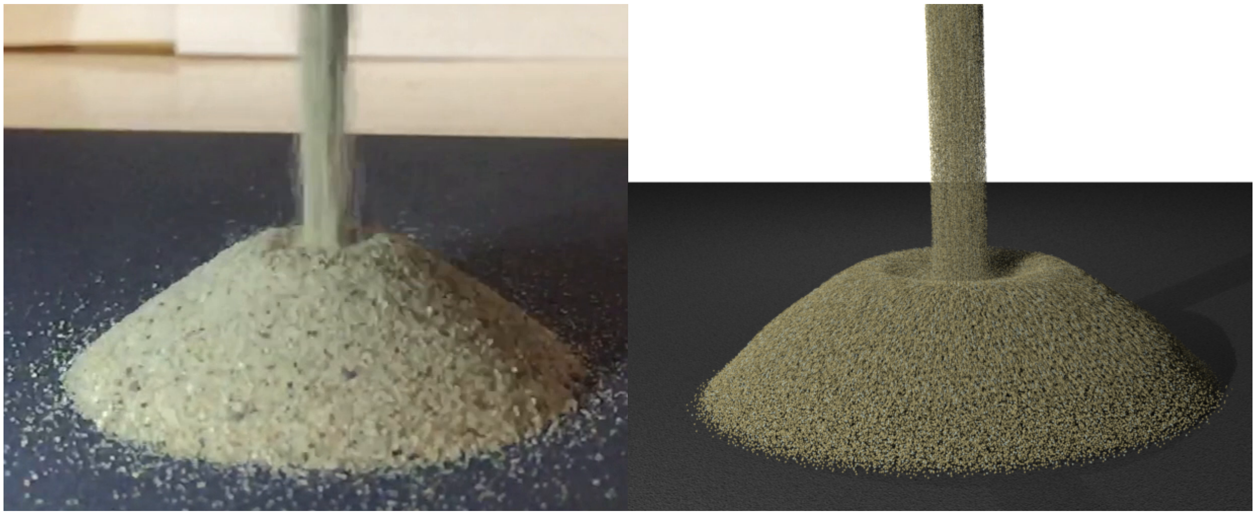

Figure 2.5: ***Avalanching.*** *Sand is poured from a spout into a pile in a lab (left) and with our method (right). Our simulation is able to capture avalanching phenomenon.*

balance is

$$R(\boldsymbol{X},0)\boldsymbol{A}(\boldsymbol{X},t) = \nabla^{\boldsymbol{X}} \cdot \boldsymbol{P}(\boldsymbol{F}(\boldsymbol{X},t)) + \boldsymbol{f}^b(\boldsymbol{X},t), \tag{2.19}$$

where $\boldsymbol{f}^b(\boldsymbol{X},t)$ represents the body force acting on the body. In this case, we are mainly interested in the force of gravity, so

$$\boldsymbol{f}^b(\boldsymbol{X},t) = R(\boldsymbol{X},0)\boldsymbol{g}, \tag{2.20}$$

with $\boldsymbol{g}$ being the gravitational acceleration constant.

2.3 Elastic material

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 $\psi(\boldsymbol{F})$.

The function $\psi(\boldsymbol{F})$ captures the essential information about the way an elastic material responds to deformation. This relationship depends on the material; we choose our model in Section 2.3.1.

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

$$\Psi = \int_{\Omega^0} \psi(\boldsymbol{F}(\boldsymbol{X}))\,\mathrm{d}\boldsymbol{X}. \tag{2.21}$$

2.3.1 Constitutive Model

For hyperelastic material, the stress tensor $\boldsymbol{P}$ is a function of the deformation gradient $\boldsymbol{F}$, and is related to the elastic energy density function $\psi(\boldsymbol{F})$ which increases with non-rigid deformation from the initial state in this way

$$\boldsymbol{P}(\boldsymbol{F}(\boldsymbol{X},t)) = \frac{\partial \psi}{\partial \boldsymbol{F}}(\boldsymbol{F}(\boldsymbol{X},t)). \tag{2.22}$$

As in Mast et al. [Mas13], we use the St. Venant-Kirchhoff model with Hencky strain energy density function to model the elastic response of sand. Given a measure of strain $\boldsymbol{\epsilon}(\boldsymbol{F})$, the St. Venant-Kirchoff energy density associated with this strain measure is defined as

$$\bar{\psi}(\boldsymbol{\epsilon}) = \frac{\lambda}{2}\operatorname{tr}(\boldsymbol{\epsilon})^2 + \mu\boldsymbol{\epsilon} : \boldsymbol{\epsilon}. \tag{2.23}$$

The measure of strain we use for sand is the Hencky strain, and it is given by

$$\boldsymbol{\epsilon}(\boldsymbol{F}) = \frac{1}{2}\log\left(\boldsymbol{F}\boldsymbol{F}^\top\right). \tag{2.24}$$

Here, we define the log of a symmetric matrix in terms of its eigenvalues. Indeed since $\boldsymbol{F}\boldsymbol{F}^\top$

is symmetric, then its eigen-decomposition can be written as $\boldsymbol{F}\boldsymbol{F}^\top = \boldsymbol{U}\boldsymbol{\Sigma}^2\boldsymbol{U}^\top$. Then

$$\log(\boldsymbol{F}\boldsymbol{F}^\top) := \boldsymbol{U}\log(\boldsymbol{\Sigma}^2)\boldsymbol{U}^\top, \tag{2.25}$$

where $\log(\boldsymbol{\Sigma}^2)$ is a diagonal matrix consisting of the logarithm of the eigenvalues of $\boldsymbol{F}\boldsymbol{F}^\top$. Hence, Equation (2.24) can be simplified to be

$$\boldsymbol{\epsilon}(\boldsymbol{F}) = \boldsymbol{U}\log\left(|\boldsymbol{\Sigma}|\right)\boldsymbol{U}^\top. \tag{2.26}$$

This combination of energy density function and measure of strain makes a number of aspects of the Drucker-Prager plastic projection to be very simple. Indeed one can prove that there is a *linear* relationship between stress and strain with this choice of modeling.

The force computation (2.86) requires the derivative of this, which is

$$\frac{\partial\psi}{\partial\boldsymbol{F}}(\boldsymbol{F}) = \boldsymbol{U}(2\mu\boldsymbol{\Sigma}^{-1}\log\boldsymbol{\Sigma} + \lambda\,\mathrm{tr}(\log\boldsymbol{\Sigma})\boldsymbol{\Sigma}^{-1})\boldsymbol{V}^\top. \tag{2.27}$$

2.4 Plasticity

In this chapter, we introduce the notion of plasticity as a mathematical model that is used to capture the behavior of granular materials. A lot of theoretical development in this chapter is taken from [BW08]. Many materials behave in a way such that it returns to a different equilibrium configuration after undergoing a large deformation, instead of its original rest configuration at the beginning. This is due to the presence of permanent inelastic deformation.

2.4.1 The Drucker-Prager yield condition

The Drucker-Prager yield condition is defined from the constraint that the shear stress should be no larger than the compressive normal stress in all directions. This expresses a mechanical interaction that is consistent with Coulomb friction. While dry sand is modeled effectively with this assumption, it precludes the effects of cohesion. However, cohesive effects can be modeled by modifying the elastic stress yield condition to be

$$c_F \operatorname{tr}(\boldsymbol{\sigma}^s) + \left\| \boldsymbol{\sigma}^s - \frac{\operatorname{tr}(\boldsymbol{\sigma}^s)}{d}\boldsymbol{I} \right\|_F \leq c_C, \tag{2.28}$$

where $\| \cdot \|_F$ denotes the Frobenius norm, $d = 2, 3$ is the spatial dimension, $c_C \geq 0$ increases with the amount of cohesion in the material and $c_F \geq 0$ increases with amount of friction between grains. Dry sand is cohesionless, and so $c_C = 0$. Inequality (2.28) is referred to as the plastic yield condition, if it is satisfied, there will be no further plastic deformation. The boundary of the region in stress space defined by $c_F \operatorname{tr}(\boldsymbol{\sigma}^s) + \|\boldsymbol{\sigma}^s - \frac{\operatorname{tr}(\boldsymbol{\sigma}^s)}{d}\boldsymbol{I}\|_F = c_C$ is called the yield surface. For states of stress on the yield surface, plastic flow will commence when a perfectly elastic assumption would drive the stress out of the region. The plasticity functions as a means of satisfying this inequality constraint.

In two dimensions, this yield condition is derived from the frictional contact between grains of sand. 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.

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

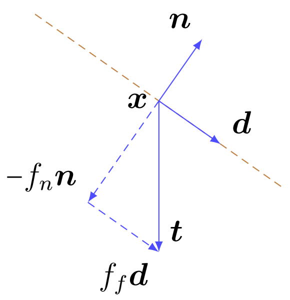

Figure 2.6: ***Coulomb friction illustration.*** *An illustration of normal and frictional force at a point $\boldsymbol{x}$ given an imaginary plane with normal $\boldsymbol{n}$.*

mechanical interactions in the material. Specifically, at point $\boldsymbol{x}$, the tensor $\boldsymbol{\sigma}(\boldsymbol{x})$ relates the force per area (or traction) $\boldsymbol{t}$ that material on one side of an imaginary plane with normal $\boldsymbol{n}$ exerts on material on the other side, as $\boldsymbol{t} = \boldsymbol{\sigma}(\boldsymbol{x})\boldsymbol{n}$. If we consider this interaction to be from friction, we can use the Coulomb model to relate the frictional force (per area) $f_f = \boldsymbol{d}^\top\boldsymbol{t}$ to the normal force (per area) $f_n = -\boldsymbol{n}^\top\boldsymbol{t}$ as $\boldsymbol{d}^\top\boldsymbol{t} \leq -\tilde{\alpha}\boldsymbol{n}^\top\boldsymbol{t}$. Here, $\boldsymbol{d}$ is the normalized projection of the traction $\boldsymbol{t}$ into the plane orthogonal to $\boldsymbol{n}$ (see Figure 2.6). In terms of $\boldsymbol{\sigma}$, this is expressed as $\boldsymbol{d}^\top\boldsymbol{\sigma}(\boldsymbol{x})\boldsymbol{n} \leq -\tilde{\alpha}\boldsymbol{n}^\top\boldsymbol{\sigma}(\boldsymbol{x})\boldsymbol{n}$.

The frictional force (per area) $f_f = \boldsymbol{d}^\top\boldsymbol{t}$ is often referred to as the shear stress (at $\boldsymbol{x}$, in direction $\boldsymbol{n}$) and the normal force (per area) is often referred to as the normal stress (at $\boldsymbol{x}$, in direction $\boldsymbol{n}$). 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 $\boldsymbol{\sigma}(\boldsymbol{x})$ as admissible (or consistent with the Coulomb model) if

$$\boldsymbol{d}^\top\boldsymbol{\sigma}(\boldsymbol{x})\boldsymbol{n} \leq -\tilde{\alpha}\boldsymbol{n}^\top\boldsymbol{\sigma}(\boldsymbol{x})\boldsymbol{n} \tag{2.29}$$

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

When the normal stress $\boldsymbol{n}^\top\boldsymbol{\sigma}(\boldsymbol{x})\boldsymbol{n}$ 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 (2.29) 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 (2.29) (for all $\boldsymbol{d}$ and $\boldsymbol{n}$) with a positive normal stress, and thus cohesion is not possible with this model.

Consider the two dimensional case and states of stress consistent with Inequality (2.29). In this case, given normal $\boldsymbol{n}$, there are only two directions $\boldsymbol{d}$ orthogonal to it, namely $\boldsymbol{d} = \pm\boldsymbol{R}\boldsymbol{n}$ where

$$\boldsymbol{R} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \tag{2.30}$$

In this case, satisfaction of Inequality (2.29) is achieved when

$$\pm\,\boldsymbol{n}^\intercal \boldsymbol{R}\boldsymbol{\sigma}(\boldsymbol{x})\boldsymbol{n} + \tilde{\alpha}\boldsymbol{n}^\intercal \boldsymbol{\sigma}(\boldsymbol{x})\boldsymbol{n} \leq 0 \tag{2.31}$$

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

$$\boldsymbol{\sigma} = \boldsymbol{Q}\boldsymbol{S}\boldsymbol{Q}^\intercal = \boldsymbol{Q}\begin{pmatrix} s_1 & 0 \\ 0 & s_2 \end{pmatrix}\boldsymbol{Q}^\intercal \tag{2.32}$$

where $\boldsymbol{Q}$ is a rotation matrix. Rewriting Inequality (2.31) in terms of the eigen-decomposition gives

$$\pm\,\boldsymbol{n}^\intercal \boldsymbol{R}\boldsymbol{Q}\boldsymbol{S}\boldsymbol{Q}^\intercal\boldsymbol{n} + \tilde{\alpha}\boldsymbol{n}^\intercal \boldsymbol{Q}\boldsymbol{S}\boldsymbol{Q}^\intercal\boldsymbol{n} \leq 0 \tag{2.33}$$

and since $\boldsymbol{R}$ and $\boldsymbol{Q}$ commute (2D rotations commute), satisfaction of Inequality (2.31) is the same as

$$\tilde{\boldsymbol{n}}^\intercal\left(\pm\boldsymbol{R}\boldsymbol{S} + \tilde{\alpha}\boldsymbol{S}\right)\tilde{\boldsymbol{n}} \leq 0 \tag{2.34}$$

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

$$\boldsymbol{R}\boldsymbol{S} = \begin{pmatrix} 0 & -s_2 \\ s_1 & 0 \end{pmatrix}. \tag{2.35}$$

Since Inequality (2.34) must be true for all $\tilde{\boldsymbol{n}}$ and choice of sign, it is equivalent to require that the maximum of

$$F(\tilde{\boldsymbol{n}}, h) = \tilde{\boldsymbol{n}}^{\intercal} \left(h\boldsymbol{RS} + \tilde{\alpha}\boldsymbol{S} \right) \tilde{\boldsymbol{n}} \tag{2.36}$$

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

$$\frac{s_1 + s_2}{2}\tilde{\alpha} + \frac{|s_1 - s_2|}{2}\sqrt{1 + \tilde{\alpha}^2}. \tag{2.37}$$

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

$$(s_1 + s_2)\frac{\tilde{\alpha}}{\sqrt{2}\sqrt{1 + \tilde{\alpha}^2}} + \frac{|s_1 - s_2|}{\sqrt{2}} \le 0,$$

which is equivalent to

$$\operatorname{tr}(\boldsymbol{\sigma}(\boldsymbol{x}))\alpha + \left\| \boldsymbol{\sigma}(\boldsymbol{x}) - \frac{\operatorname{tr}(\boldsymbol{\sigma}(\boldsymbol{x}))}{2}\boldsymbol{I} \right\|_F \le 0, \tag{2.38}$$

with $\alpha = \frac{\tilde{\alpha}}{\sqrt{2}\sqrt{1+\tilde{\alpha}^2}}$.

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 (2.38) that works for both two and three dimensions given by

$$\operatorname{tr}(\boldsymbol{\sigma}(\boldsymbol{x}))\alpha + \left\| \boldsymbol{\sigma}(\boldsymbol{x}) - \frac{\operatorname{tr}(\boldsymbol{\sigma}(\boldsymbol{x}))}{d}\boldsymbol{I} \right\|_F \le 0. \tag{2.39}$$

where d is the number of space dimension. The Drucker-Prager model uses Inequality (2.39) 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 three dimensions, the Drucker-Prager yield surface takes the shape of a cone. In summary, the Drucker-Prager model for the stress field $\boldsymbol{\sigma}$ requires that

$$y(\boldsymbol{\sigma}(\boldsymbol{x})) \le 0 \tag{2.40}$$

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

2.4.2 Stress in the presence of plasticity

As mentioned before, 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. Thus, when computing the stress from the energy density function, we need to take the effects of plasticity into account. Therefore, we define

$$\hat{\psi}(\boldsymbol{F}) := \psi(\boldsymbol{F}^E) = \psi\left(\boldsymbol{F}(\boldsymbol{F}^P)^{-1}\right), \tag{2.41}$$

where $\psi(\boldsymbol{F}^E)$ is the elastic energy density designed to penalize non-rigid $\boldsymbol{F}^E$. The first Piola-Kirchhoff stress is then given by

$$\boldsymbol{P}(\boldsymbol{F}^E(\boldsymbol{X},t), \boldsymbol{F}^P(\boldsymbol{X},t)) = \frac{\partial \psi}{\partial \boldsymbol{F}^E}(\boldsymbol{F}^E(\boldsymbol{X},t))(\boldsymbol{F}^P(\boldsymbol{X},t))^{-\top}. \tag{2.42}$$

The Cauchy stress is then

$$\boldsymbol{\sigma} = \frac{1}{\det(\boldsymbol{F})}\frac{\partial \psi}{\partial \boldsymbol{F}^E}\left(\boldsymbol{F}^E\right)\boldsymbol{F}^{E^\top}. \tag{2.43}$$

2.4.3 Unilateral hyperelasticity

In light of the Drucker-Prager cone, we propose a novel modifications that allow for more efficient implicit time integration. We note that with the plastic yield condition from Inequality (2.28), much of the energy landscape has no effect since it produces stresses outside of the yield surface. To mitigate this we modify the energy density such that it smoothly transitions to zero in these regions. This improves the performance of our semi-implicit time stepping. Essentially, our modification modifies the elastic behavior outside the yield surface

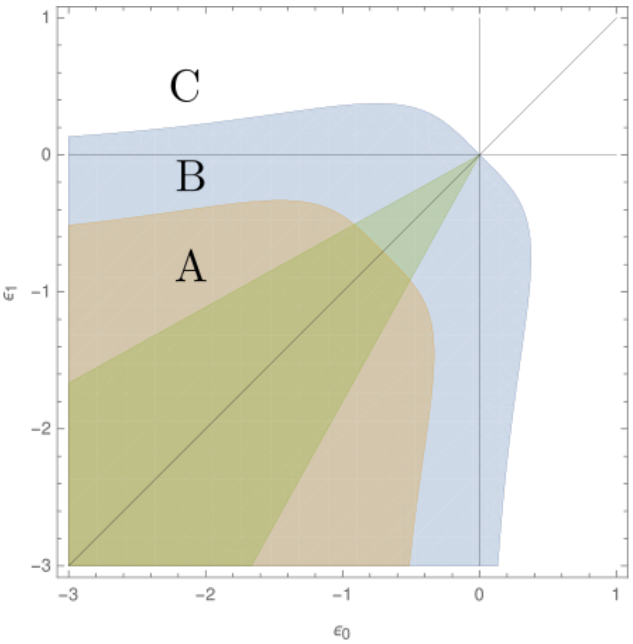

Figure 2.7: ***Multiplier region****. The contour plot of the multiplier in strain space. Region A is where the multiplier is equal to* 1*, region C is where the multiplier is* 0*, and region B is where the multiplier transitions from* 1 *to* 0 *in C^2 manner. The green cone denotes the Drucker-Prager cone in principal strain space.*

to better resemble the effects of plasticity. Our approach is similar to, and indeed inspired by the unilateral approaches in [AO11, IWT13, NGL10, DB16]. We thus refer to our modified energy as the unilateral energy function $\tilde{\psi}^U(\boldsymbol{\epsilon})$. We define this as the product of the original energy function $\tilde{\psi}(\boldsymbol{\epsilon})$ and a multiplier $h(\boldsymbol{\epsilon})$

$$\tilde{\psi}^U(\boldsymbol{\epsilon}) = \tilde{\psi}(\boldsymbol{\epsilon})h(\boldsymbol{\epsilon}). \tag{2.44}$$

The multiplier makes sure that the energy density function to transition to zero in regions outside of the yield surface (as smoothly as possible). We define it to be symmetric about the hydrostatic axis in the principal strain space (axis of equal strain, see Figure 2.7) since the original energy $\tilde{\psi}$ has this property and we wish to preserve it in $\tilde{\psi}^U$. To construct it, we partition strain space into the three regions labeled A, B, and C in Figure 2.7. In region A, the value of the multiplier is simply one and the modified constitutive model is identical to the original one. Conversely, in region C, the multiplier is set to zero. In region B, the multiplier transitions from one to zero in a C^2 manner. The boundary of region A and B

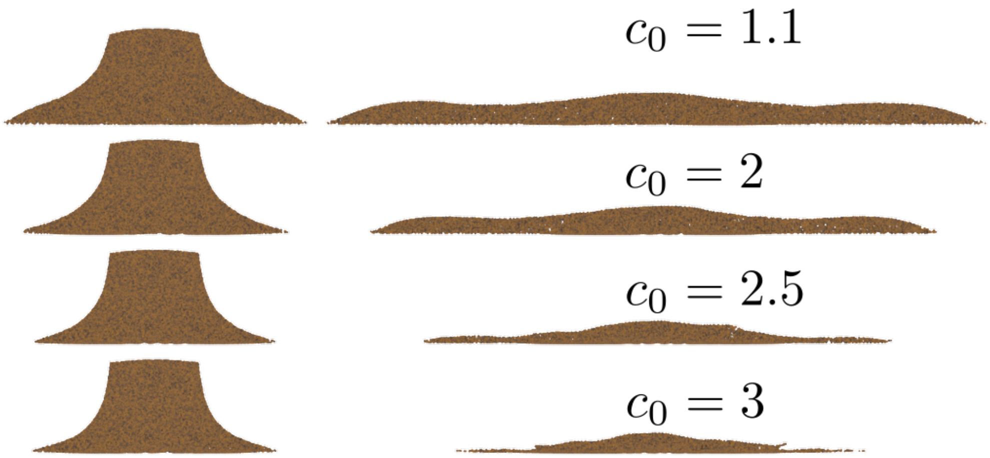

Figure 2.8: ***Unilateral extension****. We demonstrate a 2D sand column collapse with different values of c_0 coefficient in the unilateral modified energy density function. From top to bottom, $c_0 = 1.1, 2, 2.5, 3$. The left image is of an early frame and the right is of a later frame. Here parameters a and b from Equation (2.47) are $a = -0.5$, and $b = 0$. As c_0 increases, the envelopes in the unilateral extension become increasingly poor approximations of the Drucker-Prager cone and we see volume loss artifacts.*

defines two envelopes that are symmetric about the hydrostatic axis. To best preserve the behavior of sand, we want region A to cover most of the region inside of the Drucker-Prager cone, illustrated as the green cone in Figure 2.7.

The construction of this multiplier function is done as a composition of two functions: h_s which is a scalar function and f which is a function of the strain $\boldsymbol{\epsilon}$. If we let $\boldsymbol{o}$ denote the hydrostatic axis ($\boldsymbol{o} = \frac{1}{\sqrt{2}}(1,1)^\top$ in 2D or $\frac{1}{\sqrt{3}}(1,1,1)^\top$ in 3D), then we can compute $u = \boldsymbol{\epsilon} \cdot \boldsymbol{o}$ to be the component of the strain in the hydrostatic axis and $v = \|\boldsymbol{\epsilon} - u\boldsymbol{o}\|$. The function f is defined as

$$f(\boldsymbol{\epsilon}) = c_o \frac{v^4}{1 + |v|^3}. \tag{2.45}$$

The coefficient c_o controls the opening of the envelope of region A and B around the hydrostatic axis. The scalar function h_s is chosen so that the multiplier function is twice continuously differentiable, and is given by

$$h_s(z) = \begin{cases} 1 & \text{if } z < 0 \\ 0 & \text{if } z > 1 \\ 1 - 10z^3 + 15z^4 - 6z^5 & \text{otherwise.} \end{cases} \tag{2.46}$$

The multiplier function is then defined for some choice of parameters a, b, and s_C.

$$h(\boldsymbol{\epsilon}) = \begin{cases} 1 & \text{if } u + f(\boldsymbol{\epsilon}) < a + s_C \\ 0 & \text{if } u + f(\boldsymbol{\epsilon}) > b + s_C \\ h_s\left(\frac{u+f(\boldsymbol{\epsilon})-a-s_C}{b-a}\right) & \text{otherwise.} \end{cases} \tag{2.47}$$

The parameter a and b determines the intersection of the hydrostatic axis with the boundary of region A and B respectively. The parameter s_C controls a shift of this multiplier region along the hydrostatic axis.

2.5 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 automatic 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.

2.5.1 Notation

To this end, it is helpful to introduce the conventions we use in our choice of notations for the discrete settings. 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 ($\boldsymbol{v}_p^n$, $\overline{\boldsymbol{x}}_i^{n+1}$). Matrices are represented by bold uppercase Latin characters or bold Greek characters ($\boldsymbol{I}$, $\hat{\boldsymbol{F}}_p^{P,n+1}$, $\boldsymbol{\sigma}$). Derivatives alter this in the usual way, so that ∇G_{ki} is a vector and $(\nabla \boldsymbol{v})_p$ 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}).

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

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

Generally, quantities that are stored on particles have indicators of time, and those that lack other adornments are state variables (α_p^n, $\boldsymbol{F}_p^{E,n}$, $\boldsymbol{x}_p^n$; not $\boldsymbol{v}_i^n$, $\hat{\boldsymbol{F}}_p^{E,n+1}$). There are two exceptions to this. We do not store the deformation gradient itself, so $\boldsymbol{F}_p^n$ 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 $\boldsymbol{x}_p^n$, velocity $\boldsymbol{v}_p^n$, and affine velocity spatial derivative $\boldsymbol{C}_p^n$, which is related to the affine momentum. The extra matrix $\boldsymbol{C}_p^n$ stored per particle is used for APIC transfers [JSS15]. This quantity approximates the spatial derivative of the grid velocity field at the end of the previous time step.

Note that while we use $\boldsymbol{F}_p^{E,n}$ to compute forces, $\boldsymbol{F}_p^{P,n}$ (the plastic components of the deformation gradient) is not directly used in the force computation. One therefore has the option of storing the pairs $\boldsymbol{F}_p^{E,n}$ and $\boldsymbol{F}_p^{P,n}$ or $\boldsymbol{F}_p^{E,n}$ and $\boldsymbol{F}_p^n$. 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 $w_{ip}^n = N(\boldsymbol{x}_p^n - \boldsymbol{x}_i^n)$, where $\boldsymbol{x}_p^n$ and $\boldsymbol{x}_i^n$ are the locations of the particle and grid node locations. We will also need the spatial derivatives of our weights, $\nabla w_{ip}^n = \nabla N(\boldsymbol{x}_p^n - \boldsymbol{x}_i^n)$, when we compute forces. We use time indices on the fixed grid node locations $\boldsymbol{x}_i^n$ to distinguish them from estimates (such as $\overline{\boldsymbol{x}}_i^{n+1}$) 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.

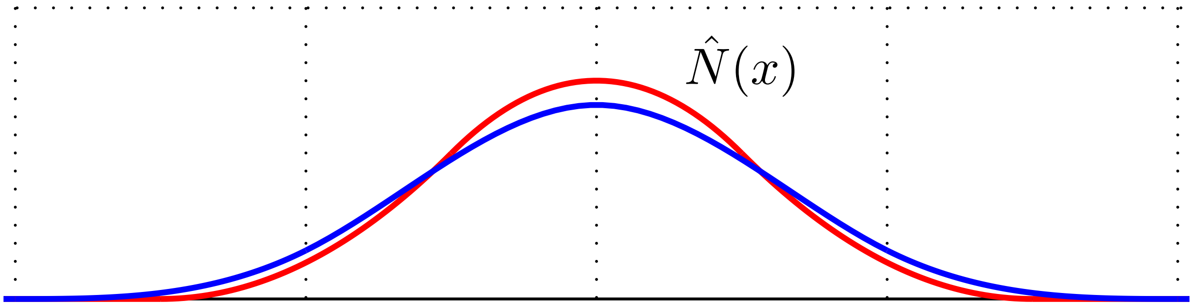

Figure 2.9: ***B-splines interpolation.*** *Cubic (blue) and quadratic (red) splines used for computing interpolation weights.*

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. The cubic B-splines is given by

$$\hat{N}(x) = \begin{cases} \frac{3}{4} - |x|^2 & 0 \le |x| < \frac{1}{2} \\ \frac{1}{2}\left(\frac{3}{2} - |x|\right)^2 & \frac{1}{2} \le |x| < \frac{3}{2} \\ 0 & \frac{3}{2} \le |x| \end{cases}, \tag{2.48}$$

and the cubic B-splines is given by

$$\hat{N}(x) = \begin{cases} \frac{1}{2}|x|^3 - |x|^2 + \frac{2}{3} & 0 \le |x| < 1 \\ \frac{1}{6}(2 - |x|)^3 & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases}. \tag{2.49}$$

We plot the quadratic and cubic kernels in Figure 2.9.

In 3D, the kernel $N_i(\cdot)$ associated with a given grid node i, whose coordinate is located at $\boldsymbol{x}_i = (x_i, y_i, z_i)$, is given by

$$N_i(\boldsymbol{x}) = N(\boldsymbol{x} - \boldsymbol{x}_i) = \hat{N}\left(\frac{x - x_i}{dx}\right)\hat{N}\left(\frac{y - y_i}{dx}\right)\hat{N}\left(\frac{z - z_i}{dx}\right), \tag{2.50}$$

where dx is the grid spacing.

2.5.2 Deriving the MPM discretization

To derive the MPM discretization of the equation to be solved, we first consider a purely elastic model (i.e. no plasticity). We also note that in this section, the use of the gradient operator ∇ should be clear from the context. To derive a weak formulation to our PDE, we take the inner product of each term in Equation (2.19) against a test function $\boldsymbol{W}(\boldsymbol{X})$ be a test function which vanishes at the boundary of Ω^0 and integrate it over Ω^0:

$$\int_{\Omega^0} R(\boldsymbol{X},0)\boldsymbol{A}(\boldsymbol{X},t^n) \cdot \boldsymbol{W}(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X} = \int_{\Omega^0} \nabla \cdot \boldsymbol{P}(\boldsymbol{X},t) \cdot \boldsymbol{W}(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X} + \int_{\Omega^0} \boldsymbol{f}^b(\boldsymbol{X},t) \cdot \boldsymbol{W}(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X}. \tag{2.51}$$

Let

$$I^{\text{inertia}} = \int_{\Omega^0} R(\boldsymbol{X},0)\boldsymbol{A}(\boldsymbol{X},t^n) \cdot \boldsymbol{W}(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X}, \tag{2.52}$$

$$I^{\text{internal}} = \int_{\Omega^0} \nabla \cdot \boldsymbol{P}(\boldsymbol{X},t) \cdot \boldsymbol{W}(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X}, \tag{2.53}$$

$$I^{\text{grav}} = \int_{\Omega^0} \boldsymbol{f}^b(\boldsymbol{X},t) \cdot \boldsymbol{W}(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X}. \tag{2.54}$$

Updated Lagrangian. The material point method can be viewed in an *updated Lagrangian* point of view. That is, we re-write the weak formulation from the original configuration at t^0 in time t^n configuration. To clean up the notation, we use Ω^n to denote Ω^{t^n}.

We also write $\boldsymbol{F}(\boldsymbol{X},t^n)$ as $\boldsymbol{F}^n(\boldsymbol{X})$ and $J(\boldsymbol{X},t^n)$ as $J^n(\boldsymbol{X})$. Furthermore, let

$$\rho^n(\boldsymbol{x},t) \coloneqq R(\boldsymbol{\varphi}^{-1}(\boldsymbol{x},t^n),t), \tag{2.55}$$

$$\boldsymbol{v}^n(\boldsymbol{x},t) \coloneqq \boldsymbol{V}(\boldsymbol{\varphi}^{-1}(\boldsymbol{x},t^n),t), \tag{2.56}$$

$$\boldsymbol{a}^n(\boldsymbol{x},t) \coloneqq \boldsymbol{A}(\boldsymbol{\varphi}^{-1}(\boldsymbol{x},t^n),t). \tag{2.57}$$

Discrete inertial term. The left hand side of the Equation (2.51) can be discretized as follow

$$I^{\text{inertia}} = \int_{\Omega^0} R(\boldsymbol{X},t)J(\boldsymbol{X},t)\boldsymbol{A}(\boldsymbol{X},t)\cdot\boldsymbol{W}(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X} \tag{2.58}$$

$$= \int_{\Omega^n} \rho^n(\boldsymbol{x},t)\boldsymbol{a}^n(\boldsymbol{x},t)\cdot\hat{\boldsymbol{w}}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \tag{2.59}$$

where we have defined $\hat{w}(\boldsymbol{x}) = W(\boldsymbol{\varphi}^{-1}(\boldsymbol{x},t^n))$. By choosing a set of basis $N_i(\boldsymbol{x})$ defined over grid nodes, we may write

$$\hat{\boldsymbol{w}}(\boldsymbol{x}) = \sum_i \boldsymbol{w}_i N_i(\boldsymbol{x}), \quad \nabla\hat{\boldsymbol{w}}(\boldsymbol{x}) = \sum_i \boldsymbol{w}_i(\nabla N_i(\boldsymbol{x}))^\intercal. \tag{2.60}$$

We also discretize the velocity in a similar way, i.e.

$$\hat{\boldsymbol{v}}^k(\boldsymbol{x}) = \sum_i \boldsymbol{v}_i^k N_i(\boldsymbol{x}), \quad \nabla\hat{\boldsymbol{v}}^k(\boldsymbol{x}) = \sum_i \boldsymbol{v}_i^k(\nabla N_i(\boldsymbol{x}))^\intercal, \tag{2.61}$$

where the superscript k denotes a discretization at time t^k. Therefore, in light of Equation (2.59), one may discretize the acceleration term as follows

$$\boldsymbol{a}^n(\boldsymbol{x},t^{n+1}) = \boldsymbol{A}(\boldsymbol{\varphi}^{-1}(\boldsymbol{x},t^n),t^{n+1}) \approx \frac{1}{\Delta t}\sum_j(\boldsymbol{v}_j^{n+1} - \boldsymbol{v}_j^n)N_j(\boldsymbol{x}_p^n), \tag{2.62}$$

where $\boldsymbol{x}_p$ denotes the position of particle p. Treating the particles as quadrature point and using index notation, we get

$$I^{\text{inertia}} \approx \frac{1}{\Delta t}\sum_p \sum_i w_{i\alpha} N_i(\boldsymbol{x}_p^n) R(\boldsymbol{X}_p, 0) \sum_j (v_{j\alpha}^{n+1} - v_{j\alpha}^n) N_j(\boldsymbol{x}_p^n) V_p^0 \tag{2.63}$$

$$= \frac{1}{\Delta t}\sum_p \sum_i w_{i\alpha} N_i(\boldsymbol{x}_p^n) m_p \sum_j (v_{j\alpha}^{n+1} - v_{j\alpha}^n) N_j(\boldsymbol{x}_p^n) \tag{2.64}$$

$$= \frac{1}{\Delta t}\sum_i w_{i\alpha} m_i \sum_j (v_{j\alpha}^{n+1} - v_{j\alpha}^n) N_j(\boldsymbol{x}_p^n), \tag{2.65}$$

where V_p^k is the volume of particle p at time t^k (so V_p^0 is the volume of particle p initially). Note that we have used Equation (2.91) for the last equality.

Force due to stress tensor. Applying the divergence theorem and assuming the free surface boundary condition, we have

$$I^{\text{internal}} = \int_{\Omega^0} (P_{\alpha\beta}(\boldsymbol{X}, t)\, W_\alpha)_{,\beta}\, (\boldsymbol{X})\, \mathrm{d}\boldsymbol{X} - \int_{\Omega^0} P_{\alpha\beta}(\boldsymbol{X}, t)\, W_{\alpha,\beta}(\boldsymbol{X})\, \mathrm{d}\boldsymbol{X} \tag{2.66}$$

$$= - \int_{\Omega^0} P_{\alpha\beta}(\boldsymbol{X}, t)\, W_{\alpha,\beta}(\boldsymbol{X})\, \mathrm{d}\boldsymbol{X}. \tag{2.67}$$

Note that

$$\frac{\partial W_\alpha}{\partial \boldsymbol{X}_\beta}(\boldsymbol{X}) = \frac{\partial \hat{w}_\alpha}{\partial \boldsymbol{x}_\gamma}(\boldsymbol{\varphi}(\boldsymbol{X}, t^n)) \frac{\partial \boldsymbol{x}_\gamma}{\partial \boldsymbol{X}_\beta}(\boldsymbol{X}, t^n) = \frac{\partial \hat{w}_\alpha}{\partial \boldsymbol{x}_\gamma}(\boldsymbol{\varphi}(\boldsymbol{X}, t^n)) \boldsymbol{F}_{\gamma\beta}(\boldsymbol{X}, t^n). \tag{2.68}$$

Hence, pulling Equation (2.67) forward to the time n configuration we may write

$$I^{\text{internal}} = - \int_{\Omega^n} J^{-1}(\boldsymbol{\varphi}^{-1}(\boldsymbol{x}, t^n)) \boldsymbol{P}_{\alpha\beta}\big(\boldsymbol{F}(\boldsymbol{\varphi}^{-1}(\boldsymbol{x}, t^n), t)\big)\, \boldsymbol{F}_{\beta\gamma}^{\top}(\boldsymbol{\varphi}^{-1}(\boldsymbol{x}, t^n), t)\, \hat{w}_{\alpha,\gamma}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}. \tag{2.69}$$

By treating the particles as quadrature point, we get

$$I^{\text{internal}} \approx - \sum_i w_{i\alpha} \sum_p P_{\alpha\beta}(\boldsymbol{F}(\boldsymbol{X}_p, t)) N_{i,\gamma}(\boldsymbol{x}_p^n) F_{p\gamma\beta}^n \frac{1}{J_p^n} V_p^n \tag{2.70}$$

$$= - \sum_i w_{i\alpha} \sum_p P_{\alpha\beta}(\boldsymbol{F}(\boldsymbol{X}_p, t)) N_{i,\gamma}(\boldsymbol{x}_p^n) F_{p\gamma\beta}^n V_p^0. \tag{2.71}$$

From Equation (2.14), we approximate $\boldsymbol{F}(\boldsymbol{X}, t^{n+1})$ as

$$\boldsymbol{F}(\boldsymbol{X}, t^{n+1}) \approx \left(\boldsymbol{I} + \Delta t \sum_j \boldsymbol{v}_j^{n+1} \nabla N_j(\boldsymbol{x}_p^n)\right) \boldsymbol{F}_p^n, \tag{2.72}$$

and so

$$I^{\text{internal}} \approx -\sum_i w_{i\alpha} \sum_p P_{\alpha\beta}\left(\left(\boldsymbol{I} + \Delta t \sum_j \boldsymbol{v}_j^{n+1} \nabla N_j(\boldsymbol{x}_p^n)\right) \boldsymbol{F}_p^n\right) N_{i,\gamma}(\boldsymbol{x}_p^n) F_{p\gamma\beta}^n V_p^0. \tag{2.73}$$

Discrete gravitational force. The body force due to gravity is the easiest one to discretize:

$$\begin{aligned} I^{\text{grav}} &= \int_{\Omega^0} R(\boldsymbol{X}, 0)\boldsymbol{g} \cdot \boldsymbol{W}(\boldsymbol{X})\, \mathrm{d}\boldsymbol{X} & (2.74)\\ &= \int_{\Omega^n} \rho^n(\boldsymbol{x}, t)\boldsymbol{g} \cdot \hat{\boldsymbol{w}}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}, & (2.75) \end{aligned}$$

and at time t^{n+1},

$$\begin{aligned} I^{\text{grav}} &\approx \sum_p \sum_i w_{i\alpha} N_i(\boldsymbol{x}_p^n) R(\boldsymbol{X}_p, 0) \sum_j \boldsymbol{g} N_j(\boldsymbol{x}_p^n) V_p^0 & (2.76)\\ &= \sum_p \sum_i w_{i\alpha} N_i(\boldsymbol{x}_p^n) m_p \boldsymbol{g}, & (2.77)\\ &= \sum_i w_{i\alpha} m_i \boldsymbol{g}, & (2.78) \end{aligned}$$

where the second equality comes from the partition of unity property due to our choice of B-splines basis functions.

Full discrete equation. Combining all of these forces together, we get

$$\begin{aligned} \sum_i w_{i\alpha} m_i \sum_j \frac{(v_{j\alpha}^{n+1} - v_{j\alpha}^n)}{\Delta t} N_j(\boldsymbol{x}_p^n) = -\sum_i w_{i\alpha} \sum_p P_{\alpha\beta}(\boldsymbol{F}(\boldsymbol{X}_p, t)) N_{i,\gamma}(\boldsymbol{x}_p^n) F_{p\gamma\beta}^n V_p^0 \\ + \sum_i w_{i\alpha} m_i \boldsymbol{g}. \end{aligned} \tag{2.79}$$

Noting that the weights $w_{i\alpha}$ can be arbitrary, the equation becomes

$$m_i \sum_j \frac{(v_{j\alpha}^{n+1} - v_{j\alpha}^n)}{\Delta t} N_j(\boldsymbol{x}_p^n) = -\sum_p P_{\alpha\beta}(\boldsymbol{F}(\boldsymbol{X}_p, t)) N_{i,\gamma}(\boldsymbol{x}_p^n) F_{p\gamma\beta}^n V_p^0 + m_i \boldsymbol{g}. \tag{2.80}$$

We can further apply mass-lumping technique to get a diagonal mass matrix, and in this case we get

$$m_i \frac{(v_{i\alpha}^{n+1} - v_{i\alpha}^n)}{\Delta t} = -\sum_p P_{\alpha\beta}(\boldsymbol{F}(\boldsymbol{X}_p, t)) N_{i,\gamma}(\boldsymbol{x}_p^n) F_{p\gamma\beta}^n V_p^0 + m_i \boldsymbol{g}. \tag{2.81}$$

When we simulate elastoplastic material, we evaluate $\boldsymbol{P}$ at $\boldsymbol{F}^E$, and the equation to be solved becomes

$$m_i^n \frac{v_{i\alpha}^{n+1} - v_{i\alpha}^n}{\Delta t} = -\sum_p P_{\alpha\beta}(\boldsymbol{F}_p^E(\boldsymbol{v}^{n+1})) N_{i,\gamma}(\boldsymbol{x}_p^n) F_{p\gamma\beta}^n V_p^0 + m_i \boldsymbol{g}. \tag{2.82}$$

2.5.3 Notes on discrete energy and force

In the previous section, we presented how forces in an MPM discretization is derived in a very similar way as an FEM discretization from a balance of linear momentum. We can also view it as a byproduct of the discrete potential energy. We can discretize the potential energy with a sum on particles, i.e.

$$\Psi = \sum_p V_p^0 \psi(\boldsymbol{F}_p^E). \tag{2.83}$$

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 $\boldsymbol{F}_p^E$ contributes to the energy [BW08].

If the state of the system is described by a finite number of positions $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ (picture a collection of point masses connected by springs), then the potential energy can be written $\Phi(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m)$. 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 $\boldsymbol{f}_j = -\frac{\partial \Psi}{\partial \boldsymbol{x}_j}$.

With MPM, grid nodes are temporarily Lagrangian, and can be moved to define the force. If the current grid node velocity is $\boldsymbol{v}_i$, then its position can be approximated as $\boldsymbol{x}_i = \boldsymbol{x}_i^n + \Delta t\, \boldsymbol{v}_i$. Considering a different ending position implies a different velocity to get there. These node velocities are used in (2.100) to compute $(\nabla \boldsymbol{v})_p$, which is in turn used by (2.101) to compute a new deformation gradient $\boldsymbol{F}_p^E$. This deformation gradient will be used to compute energy density using a model $\psi(\boldsymbol{F}_p^E)$, 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:

$$\Psi(\boldsymbol{x}_i) = \sum_p V_p^0 \psi(\boldsymbol{F}_p^E(\boldsymbol{x}_i)), \tag{2.84}$$

with

$$\boldsymbol{F}_p^E(\boldsymbol{x}_i) = \left(\boldsymbol{I} + \sum_i (\boldsymbol{x}_i - \boldsymbol{x}_i^n)(\nabla w_{ip}^n)^\top\right)\boldsymbol{F}_p^{E,n}. \tag{2.85}$$

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

$$\boldsymbol{f}_i(\boldsymbol{F}_p^E) = -\frac{\partial \Psi}{\partial \boldsymbol{x}_i} = -\sum_p V_p^0 \left(\frac{\partial \psi}{\partial \boldsymbol{F}}(\boldsymbol{F}_p^E)\right)(\boldsymbol{F}_p^{E,n})^\top \nabla w_{ip}^n. \tag{2.86}$$

Note that $\boldsymbol{F}_p^E$ is the function parameter but $\boldsymbol{F}_p^{E,n}$ 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].

2.5.4 Plasticity in discrete setting

Here, we describe our function $\boldsymbol{Z}(\boldsymbol{F}_p^E, \alpha_p)$, which projects the deformation gradient $\boldsymbol{F}_p^E$ to the yield surface defined by the parameter α_p, which controls hardening, for example. As mentioned before, in the space of principal stress, the yield surface looks like a cone (see

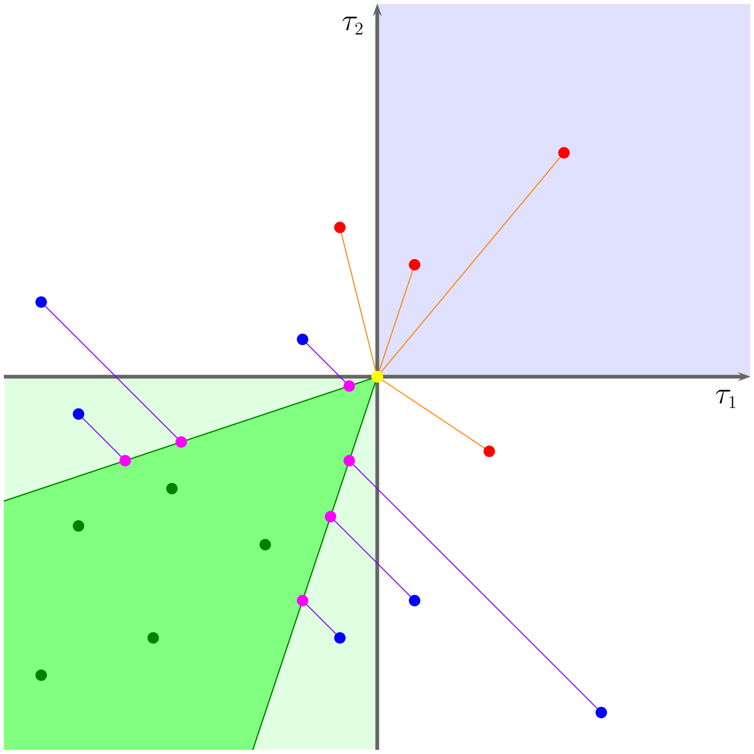

Figure 2.10: ***Drucker-Prager region.*** *The yield surface for Drucker-Prager is shown in principal stretch space. The yield surface has the shape of a cone with its tip at the origin, which corresponds to no stress. Green particles are inside the yield surface and exhibit an elastic response. Blue 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.*

Figure 2.10). There are three possible cases that must be considered. If the stress lies within the yield 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 should project to the side of the cone. Examples of these cases in an actual simulation can be seen in Figure 2.11.

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

$$\hat{\boldsymbol{\epsilon}}_p = \boldsymbol{\epsilon}_p - \frac{\operatorname{tr}(\boldsymbol{\epsilon}_p)}{d}\boldsymbol{I}, \qquad \delta\gamma_p = \|\hat{\boldsymbol{\epsilon}}_p\|_F + \frac{d\lambda + 2\mu}{2\mu}\operatorname{tr}(\boldsymbol{\epsilon}_p)\alpha_p, \tag{2.87}$$

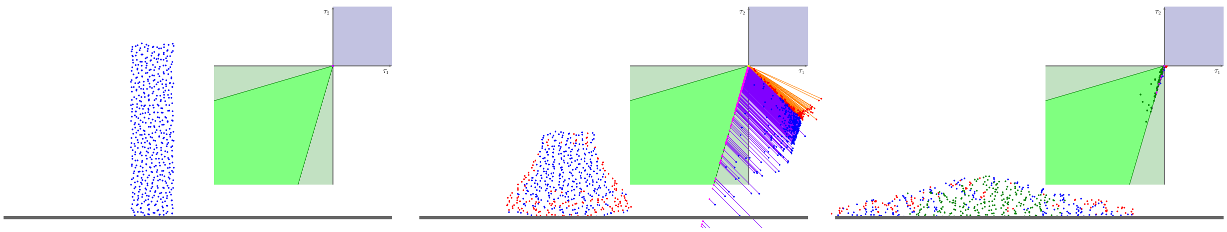

Figure 2.11: ***Sand column collapse.*** *A column of sand collapses into a pile. Sand particles are colored based on their current plastic deformation behavior. The plot shows the locations of these particles in principal stress space. Green particles lie within the yield surface and experience no plasticity. 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.*

where d is the spatial dimension and $\delta\gamma_p$ is the amount of plastic deformation. If $\delta\gamma_p \leq 0$, then the candidate $\boldsymbol{F}_p^E$ is already in the yield surface and should be returned without modification (Case I). If $\|\hat{\boldsymbol{\epsilon}}_p\|_F = 0$ or $\operatorname{tr}(\boldsymbol{\epsilon}_p) > 0$, then we need to project to the cone's tip (Case II), in which case we should return $\boldsymbol{U}_p\boldsymbol{V}_p{}^\top$. Otherwise, we should project to the cone surface (Case III) by returning $\boldsymbol{U}_p e^{\boldsymbol{H}_p}\boldsymbol{V}_p{}^\top$, where

$$\boldsymbol{H}_p = \boldsymbol{\epsilon}_p - \delta\gamma_p \frac{\hat{\boldsymbol{\epsilon}}_p}{\|\hat{\boldsymbol{\epsilon}}_p\|_F}. \tag{2.88}$$

Note that the operations $\log \boldsymbol{\Sigma}_p$ and $e^{\boldsymbol{H}_p}$ 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 $\boldsymbol{Z}$ has a straightforward singular value decomposition ($\boldsymbol{U}_p$ and $\boldsymbol{V}_p$ do not change), and this decomposition will be required when computing the force. We avoid the extra decomposition by returning the diagonal part ($\boldsymbol{\Sigma}_p$, $\boldsymbol{I}$, or $e^{\boldsymbol{H}_p}$) rather than the full result ($\boldsymbol{F}_p^E$, $\boldsymbol{U}_p\boldsymbol{V}_p{}^\top$, or $\boldsymbol{U}_p e^{\boldsymbol{H}_p}\boldsymbol{V}_p{}^\top$).

2.5.5 Notes on volume change in projection

We present two ways of dealing with the volume change that might occur in during the projection step when we apply plasticity. The method described so far has the desirable feature that sand is prevented from compressing arbitrarily as a byproduct of losing volume

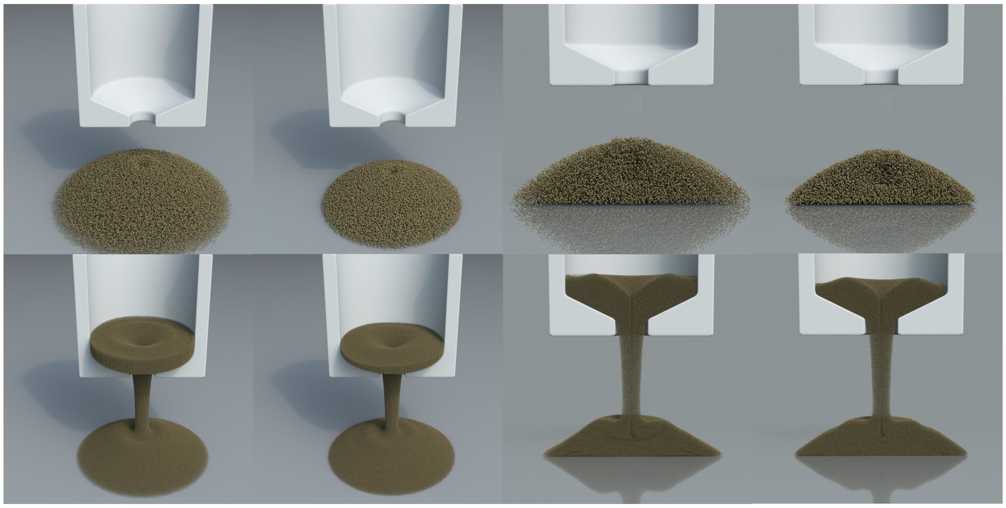

Figure 2.12: ***Pouring sand***. *The top row depicts the result of an explicit simulation with a coarser grid size ($dx = 10^{-2}$), while the bottom row corresponds to a finer grid size ($dx = 10^{-3}$). The first and third column correspond to a projection step without volume correction, while the second and fourth column uses our volume correction algorithm.*

in the plasticity projection. To see this, note that a change in $\det(\boldsymbol{F}_p^E)$ corresponds to a change in volume of the elastic deformation. In Case I, $\boldsymbol{F}_p^E$ 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 $\operatorname{tr}(\hat{\boldsymbol{\epsilon}}_p) = 0$, the Drucker-Prager model uses a non-associative flow to preserve volume during the plastic projection

$$\det(\boldsymbol{U}_p e^{\boldsymbol{H}_p} \boldsymbol{V}_p^\top) = e^{\operatorname{tr}(\boldsymbol{H}_p)} = e^{\operatorname{tr}(\boldsymbol{\epsilon}_p)} = \det(\boldsymbol{\Sigma}_p) = \det(\boldsymbol{F}_p^E).$$

The key to retaining volume in this case is to ensure that $\operatorname{tr}(\boldsymbol{H}_p) = \operatorname{tr}(\boldsymbol{\epsilon}_p)$, 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.

However, while this approach is adequate for projection directions perpendicular to the hydrostatic axis (Case III), (elastic) deformation gradient which is projected to the tip of the Drucker-Prager cone (case II) can induce volume gain. This occurs when a particle undergoes expansion that induces a cohesive elastic stress. In this case, stress is projected to the tip,

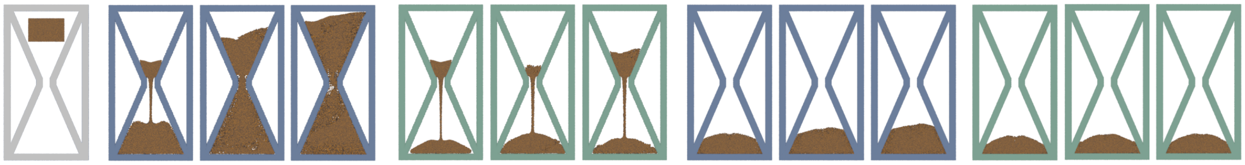

Figure 2.13: ***2D hourglass.*** *An hourglass is flipped two times. The left most hourglass shaded grey depicts the initial state of sand at the beginning of the simulation. Each set-of-three figures depicts the state of the sand after going through the neck the first time, and after flipping the hourglass for the first and second time. The first two sets and the last two sets have grid* $dx = 7.4 \times 10^{-3}$ *and* $dx = 3.7 \times 10^{-3}$ *respectively. The first and third set are run without any volume correction fix, while the second and last set are run with our volume fix.*

which is a stress free state. The particle is then in a new rest state and any motion that would return it to its initial volume would be penalized elastically. This phenomena can lead to some clearly non-physical behavior (see Figure 2.13) and there are existing corrections to this in the mechanics literature, e.g. Dunatunga and Kamrin [DK15] model material as a disconnected stress-free medium under sufficient expansion.

We combat this artifact by giving each sand particle an extra scalar attribute v_{cp} which tracks changes in the log of the volume gained during extension. This can be naturally taken into account in the logarithmic-strain-based constitutive model to allow for compression in the event of prior net expansion.

Since we need to keep track of v_{cp} it is useful to discuss this step in the context of a time-step, i.e. going from time t^n to t^{n+1}. Let $\hat{\boldsymbol{F}}_p^{E,n+1}$ be defined as in Equation (2.101). This is the part of the deformation gradient that needs to be projected. When we perform the Drucker-Prager projection, we update $\boldsymbol{H}^{n+1}$ according to

$$\boldsymbol{H}_p^{n+1} = \boldsymbol{Z}\left(\boldsymbol{\epsilon}_p + \frac{v_{cp}^n}{d}\boldsymbol{I}\right), \tag{2.89}$$

where $\boldsymbol{Z}$ is the projection operator described above. This can be interpreted as projecting

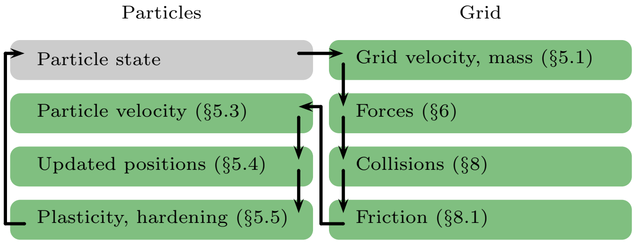

Figure 2.14: ***Overview of MPM stages.*** *Breakdowns of an MPM algorithm.*

the elastic strain plus the volume gain term. The quantity v_{cp} is updated according to

$$v_{cp}^{n+1} = v_{cp}^{n} + \log\left(\det\left(\boldsymbol{F}^{E,n+1}\right)\right) - \log\left(\det\left(\hat{\boldsymbol{F}}^{E,n+1}\right)\right), \tag{2.90}$$

where $v_{cp}^0 = 0$. Lastly, we update $\boldsymbol{F}^{E,n+1} = \boldsymbol{U} e^{\boldsymbol{H}_p^{n+1}} \boldsymbol{V}^\top$. See Figures 2.12 and 2.13 for a demonstration of this effect. We note that this projection is particularly defined for constitutive models written in terms of the logarithmic strain. For a more general constitutive model, it would require non-trivial modification.

2.6 Algorithm

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 2.14.

1. **Transfer to grid.** Transfer mass and momentum from particles to the grid. Use mass and momentum to compute velocity on the grid (§2.6.1).

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 (§2.5.3).

3. **Grid collisions.** Project grid velocities for collisions against scripted bodies and obstacles, ignoring friction (§2.6.9). For implicit, this is merged with the force application step (§2.6.7).

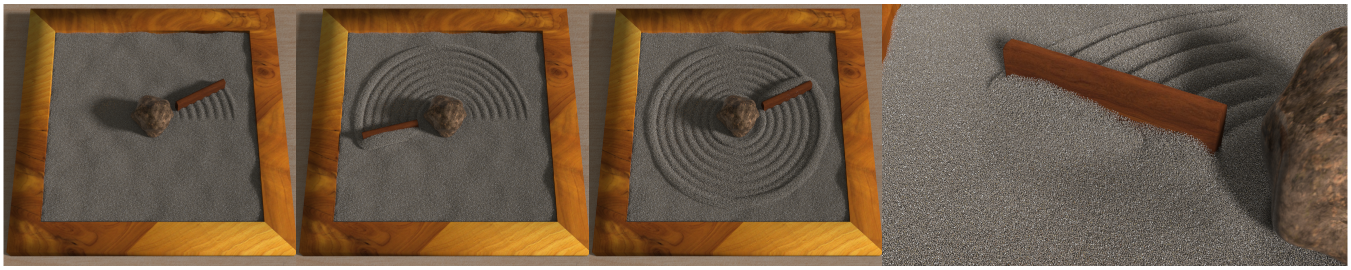

Figure 2.15: ***Raking.*** *A rake is dragged around a rock, producing a circular pattern in the sand.*

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 (§2.6.10).
5. **Transfer to particles.** Transfer velocities from grid to particles, being careful to handle friction in a manner that does not lead to inconsistencies (§2.6.3).
6. **Update particles.** Update remaining particle state, including positions and deformation gradient (§2.6.4).
7. **Plasticity and hardening.** Project the deformation gradient for plasticity, updating the elastic and plastic parts. Perform hardening, which updates the plastic yield surface (§2.6.5).

2.6.1 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:

$$m_i^n = \sum_p w_{ip}^n m_p. \tag{2.91}$$

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 $\boldsymbol{v}_p^n$ and $\boldsymbol{C}_p^n$. The velocity spatial derivative

$\boldsymbol{C}_p^n$ is related to the affine momentum $\boldsymbol{B}_p^n$ through $\boldsymbol{C}_p^n = \boldsymbol{B}_p^n(\boldsymbol{D}_p^n)^{-1}$, where $\boldsymbol{D}_p^n$ is a matrix that behaves as an inertia tensor and is

$$\boldsymbol{D}_p^n = \sum_i w_{ip}^n(\boldsymbol{x}_i^n - \boldsymbol{x}_p^n)(\boldsymbol{x}_i^n - \boldsymbol{x}_p^n)^\top = \begin{cases} \frac{dx^2}{3}\boldsymbol{I} & \text{cubic} \\ \frac{dx^2}{4}\boldsymbol{I} & \text{quadratic} \end{cases} \tag{2.92}$$

where dx is the grid spacing. Although the definition of the inertia tensor $\boldsymbol{D}_p^n$ 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 $\boldsymbol{C}_p^n$, we can define an affine velocity field $\overline{\boldsymbol{v}}_p(\boldsymbol{x})$ for particle p by $\overline{\boldsymbol{v}}_p(\boldsymbol{x}) = \boldsymbol{v}_p + \boldsymbol{C}_p^n(\boldsymbol{x} - \boldsymbol{x}_p^n)$. The momentum contribution from particle p to node i is $w_{ip}^n m_p \overline{\boldsymbol{v}}_p(\boldsymbol{x}_i^n)$. This leads to the full form of the velocity transfer, namely

$$\boldsymbol{v}_i^n = \frac{1}{m_i^n}\sum_p w_{ip}^n m_p(\boldsymbol{v}_p^n + \boldsymbol{C}_p^n(\boldsymbol{x}_i^n - \boldsymbol{x}_p^n)). \tag{2.93}$$

2.6.2 Grid update

We next update velocities on the grid. This involves applying forces and processing for collisions with scripted objects. We present three approaches for doing this, explicit, implicit, and semi-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 discuss the implicit formulation in Section 2.6.7 and the semi-implicit formulation in Section 2.6.8.

The simplest approach for handling forces is explicit. In this case, we compute and apply an explicit force as described in Section 2.5.3, namely

$$\boldsymbol{v}_i^\star = \boldsymbol{v}_i^n + \frac{\Delta t}{m_i^n}\boldsymbol{f}_i(\boldsymbol{F}_p^{E,n}). \tag{2.94}$$

After forces are applied, we can process the velocities for collisions $\boldsymbol{v}_i^\star \to \overline{\boldsymbol{v}}_i^{n+1}$ and then apply

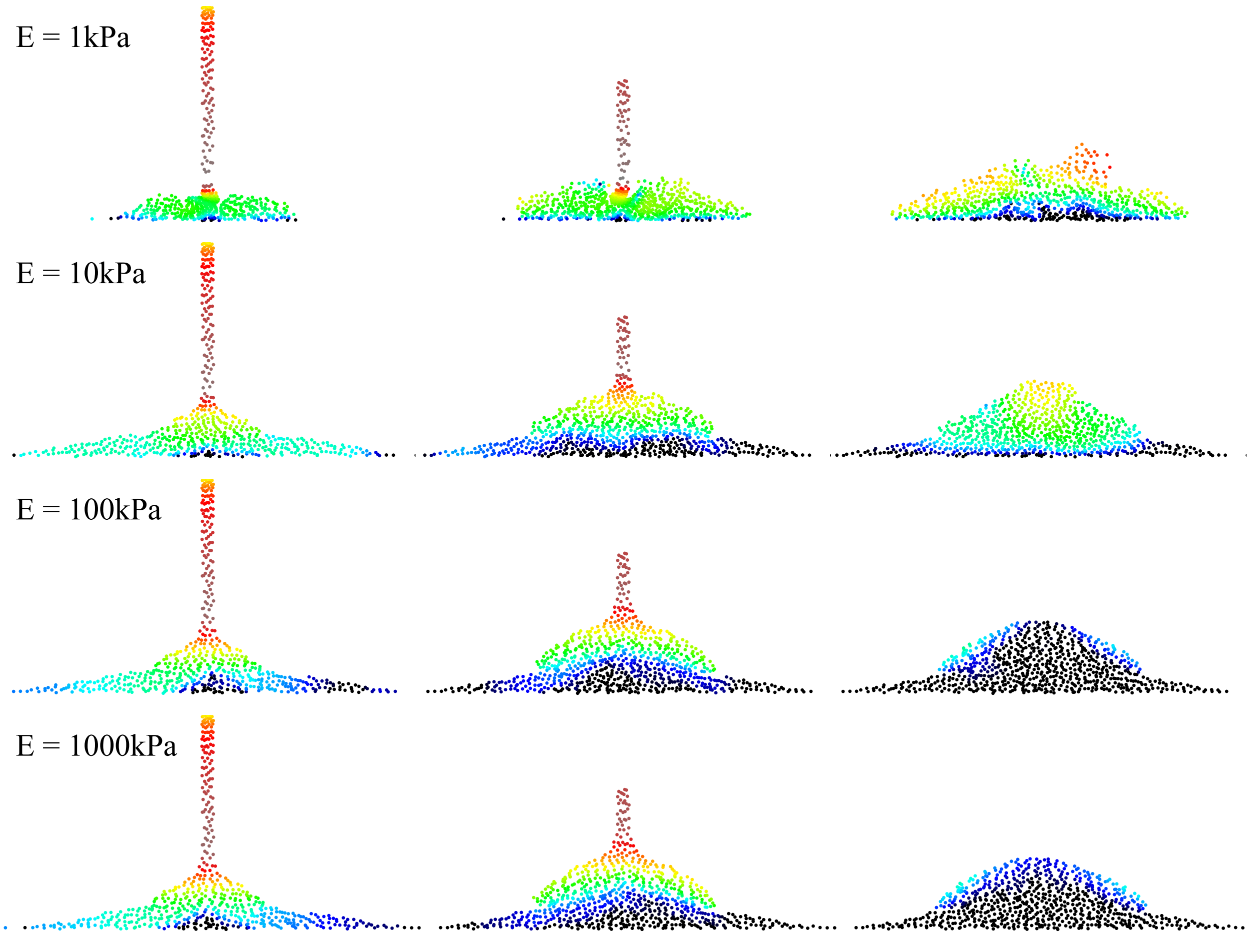

Figure 2.16: ***Effects of Young's modulus.*** *This simulation shows the effects of Young's modulus on the behavior of a simulation. Sand with a very low Young's modulus tends to be bouncy. The behavior is more like sand as the Young's modulus approaches its physical value.*

friction $\overline{\boldsymbol{v}}_i^{n+1} \to \tilde{\boldsymbol{v}}_i^{n+1}$. The collision processing is described in Section 2.6.9.

2.6.3 Transfer to particles

Next we transfer velocities from the grid back to particles. Since we are using APIC, we need to compute new velocities $\boldsymbol{v}_p^{n+1}$ and the velocity spatial derivative $\boldsymbol{C}_p^{n+1}$. Velocities are interpolated back to particles in the straightforward way

$$\boldsymbol{v}_p^{n+1} = \sum_i w_{ip}^n \tilde{\boldsymbol{v}}_i^{n+1}. \tag{2.95}$$

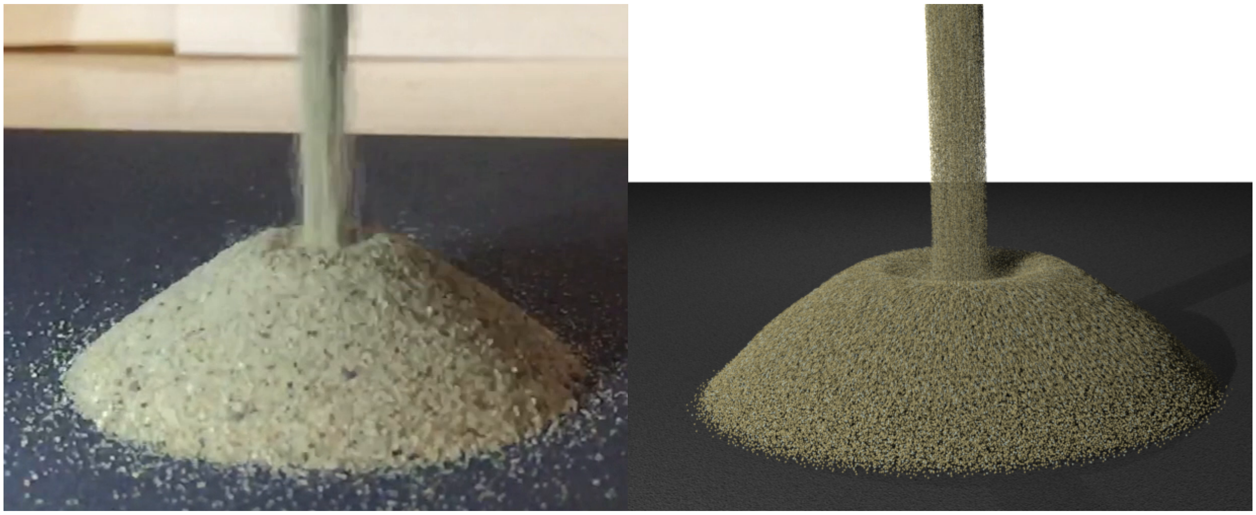

Figure 2.17: ***Avalanching.*** *Sand is poured from a spout into a pile in a lab (left) and with our method (right). Our simulation is able to capture avalanching phenomenon.*

The velocity spatial derivative is updated according to

$$\boldsymbol{C}_p^{n+1} = \sum_i w_{ip}^n \tilde{\boldsymbol{v}}_i^{n+1} \left((\boldsymbol{D}_p^n)^{-1} \left(\boldsymbol{x}_i^n - \boldsymbol{x}_p^n \right) \right)^\top , \tag{2.96}$$

where the matrix $\boldsymbol{D}_p^n$ is defined in Equation (2.92). If we choose to work with the affine momentum matrix, then the transfer for $\boldsymbol{B}_p^{n+1}$ is

$$\boldsymbol{B}_p^{n+1} = \sum_i w_{ip}^n \tilde{\boldsymbol{v}}_i^{n+1} (\boldsymbol{x}_i^n - \boldsymbol{x}_p^n)^\top . \tag{2.97}$$

2.6.4 Update particle state

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

$$\boldsymbol{x}_p^{n+1} = \sum_i w_{ip}^n \overline{\boldsymbol{x}}_i^{n+1} . \tag{2.98}$$

Since the particles move with the flow, the material derivative in Equation (2.14) is just a normal time derivative and a simple difference yields the particle deformation gradient

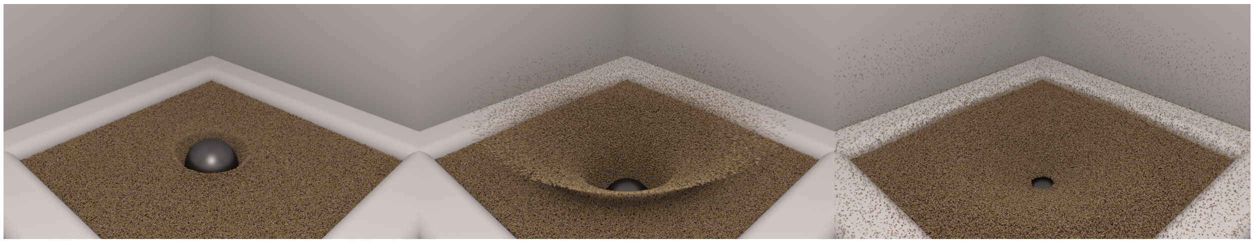

Figure 2.18: ***Coupling with stiff material.*** *A solid ball drops into a sandbox, spraying sand in all directions.*

update

$$\boldsymbol{F}_p^{n+1} = \boldsymbol{F}_p^n + \Delta t \, (\nabla \boldsymbol{v})_p \boldsymbol{F}_p^n, \tag{2.99}$$

where $(\nabla \boldsymbol{v})_p$ is calculated by differentiating (2.95)

$$(\nabla \boldsymbol{v})_p = \sum_i \overline{\boldsymbol{v}}_i^{n+1} (\nabla w_{ip}^n)^\top. \tag{2.100}$$

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

$$\hat{\boldsymbol{F}}_p^{E,n+1} = \boldsymbol{F}_p^{E,n} + \Delta t \, (\nabla \boldsymbol{v})_p \boldsymbol{F}_p^{E,n}. \tag{2.101}$$

The plastic update is covered in Section 2.5.4.

Note that the update of the particle position and deformation gradient use $\overline{\boldsymbol{v}}_i^{n+1}$ while the velocity and related quantities use $\tilde{\boldsymbol{v}}_i^{n+1}$. 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, $\tilde{\boldsymbol{v}}_i^{n+1}$ could be used for both position and velocity related updates.

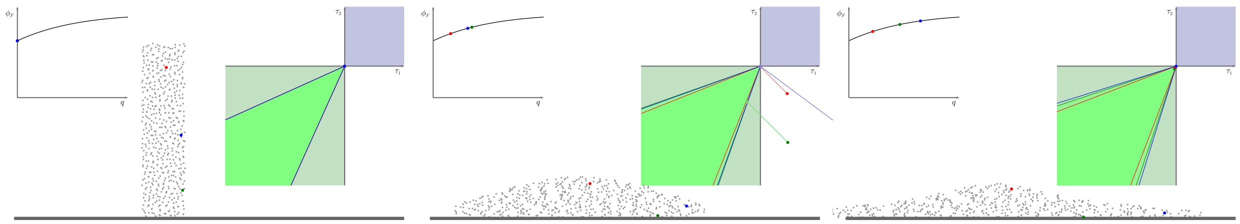

Figure 2.19: ***Hardening.*** *Three particles in a collapsing pile of sand are colored for reference. As these particles deform plasticly, their yield surface changes as they undergo hardening, resulting in a wider cone for projection. Hardening causes each particle to have its own yield surface.*

2.6.5 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 $\boldsymbol{Z}(\cdot,\cdot)$ which we have described in detail in Section 2.5.4. Plasticity does not change the full deformation gradient, so that $\boldsymbol{F}_p^{n+1} = \hat{\boldsymbol{F}}_p^{E,n+1}\hat{\boldsymbol{F}}_p^{P,n+1} = \boldsymbol{F}_p^{E,n+1}\boldsymbol{F}_p^{P,n+1}$. This allows us to update the plastic part.

$$\boldsymbol{F}_p^{E,n+1} = \boldsymbol{Z}(\hat{\boldsymbol{F}}_p^{E,n+1}, \alpha_p^n) \tag{2.102}$$

$$\boldsymbol{F}_p^{P,n+1} = (\boldsymbol{F}_p^{E,n+1})^{-1}\hat{\boldsymbol{F}}_p^{E,n+1}\hat{\boldsymbol{F}}_p^{P,n+1} \tag{2.103}$$

The last step is hardening which updates $\alpha_p^n \to \alpha_p^{n+1}$ which is described in Section 2.6.6.

2.6.6 Hardening

We adopt the hardening model of Mast et al. [MAM14a], 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 $\delta q_p = \|\boldsymbol{\epsilon}_p^{E,n+1}\|_F$. 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

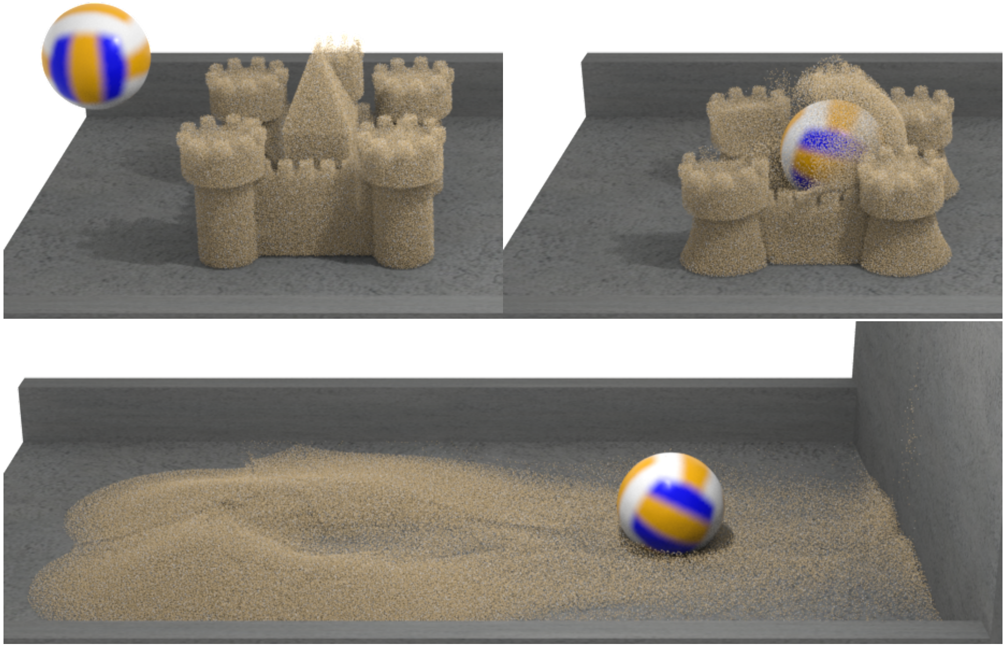

Figure 2.20: ***Coupling with deformable object.*** *A sand castle is hit with a deformable ball while falling. The sand and ball are fully coupled in the simulation.*

update using

$$q_p^{n+1} = q_p^n + \delta q_p \tag{2.104}$$

$$\phi_{F_p} = h_0 + (h_1 q_p^{n+1} - h_3) e^{-h_2 q_p^{n+1}} \tag{2.105}$$

$$\alpha_p^{n+1} = \sqrt{\frac{2}{3}} \frac{2 \sin \phi_{F_p}}{3 - \sin \phi_{F_p}} \tag{2.106}$$

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 (2.105) 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 $h_1, h_2 \geq 0$. Figure 2.19 illustrates the change in yield surface as particles undergo hardening in 2D.

2.6.7 Implicit velocity update

The implicit velocity update is

$$\overline{\boldsymbol{v}}_i^{n+1} = \boldsymbol{v}_i^n + \frac{\Delta t}{m_i^n}\boldsymbol{f}_i(\boldsymbol{F}_p^{E,n+1}) + \sum_k \nabla G_{ki}\lambda_k \tag{2.107}$$

subject to the additional conditions $G_k \geq 0$, $\lambda_k \geq 0$, and $G_k\lambda_k = 0$. Here, $G_k(\overline{\boldsymbol{x}}_i^{n+1}) \geq 0$, with $\overline{\boldsymbol{x}}_i^{n+1} = \boldsymbol{x}_i^n + \Delta t \overline{\boldsymbol{v}}_i^{n+1}$, is the collision-free criterion for all object-node collision pairs k. (§2.6.9) These forces are implicit, since $\boldsymbol{F}_p^{E,n+1}$ depends on $\overline{\boldsymbol{v}}_i^{n+1}$ through (2.101), (2.100), and (2.102). As in the explicit case, we complete the grid update by applying friction $\overline{\boldsymbol{v}}_i^{n+1} \rightarrow \tilde{\boldsymbol{v}}_i^{n+1}$ and described in Section 2.6.9.

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.

2.6.8 Semi implicit velocity update

This time-stepping scheme for sand is used in [PGK17]. This is also used by Stomakhin et al. [SSC13] for simulation of snow. In a semi-implicit time-stepping, in going from time t^n to t^{n+1}, one assumes that the material only has an elastic response in an implicit step to solve for the velocity at time $\boldsymbol{v}^{n+1}$. Plasticity is then used as a post-process to redistribute the elastic component and the plastic component of the deformations ($\boldsymbol{F}^E$ and $\boldsymbol{F}^P$). Whenever

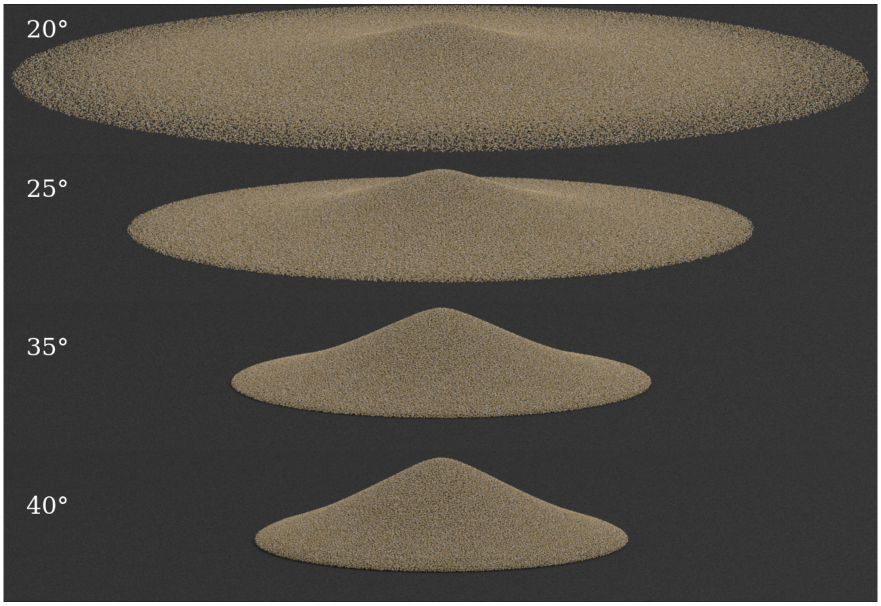

Figure 2.21: ***Effects of friction angle on piling.*** *Varying the friction angle changes the shape of a pile of sand. A larger angle produces a taller sand pile with steeper sides.*

we are simulating dry sand with a semi-implicit time-stepping, we use the unilateral energy density function as described in Section 2.4.3.

2.6.9 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 $\phi(\boldsymbol{x})$ 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 $\phi(\boldsymbol{x}_p^{n+1}) \geq 0$. In practice, processing collisions directly on particles produces poor results, since it causes $\boldsymbol{x}_p^{n+1}$ and $\boldsymbol{F}_p^{E,n+1}$ 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 $\boldsymbol{x}_p^{n+1}$ will be computed based on $\overline{\boldsymbol{v}}_i^{n+1}$, one could adjust $\overline{\boldsymbol{v}}_i^{n+1}$ to enforce $\phi(\boldsymbol{x}_p^{n+1}) \geq 0$.

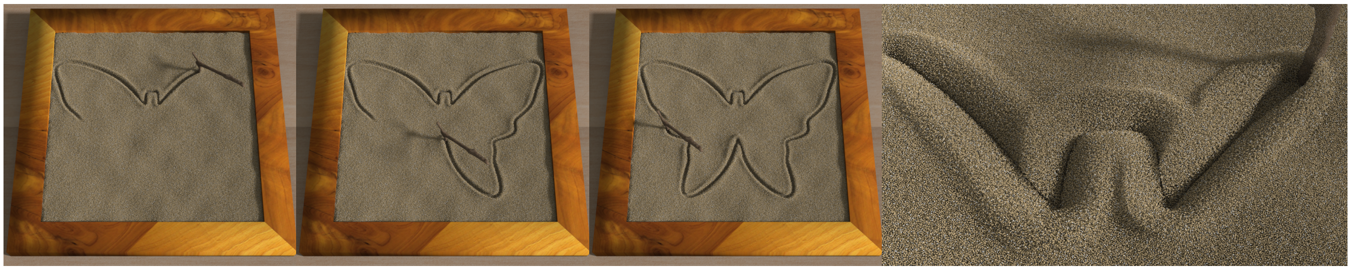

Figure 2.22: ***Drawing in sand.*** *A stick is dragged through a bed of sand, tracing out a butterfly shape in the sand.*

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 $\phi(\overline{\boldsymbol{x}}_i^{n+1}) \geq 0$ does not make sense. There *should* be grid nodes inside obstacles. A set of constraints $G_k(\overline{\boldsymbol{x}}_i^{n+1}) \geq 0$ or $G_k(\overline{\boldsymbol{x}}_i^{n+1}) = 0$ 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 $\overline{\boldsymbol{v}}_i^{n+1} = \boldsymbol{v}_b^{n+1}$, where $\boldsymbol{v}_b^{n+1}$ is the velocity of the collision object at the candidate position. In terms of positions, this is $G_k(\overline{\boldsymbol{x}}_i^{n+1}) = \overline{\boldsymbol{x}}_i^{n+1} - \boldsymbol{x}_i^n - \Delta t\, \boldsymbol{v}_b^{n+1} = 0$. 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 ($\phi(\boldsymbol{x}_i^n) < 0$), then it should not penetrate any deeper, $\phi(\overline{\boldsymbol{x}}_i^{n+1}) \geq \phi(\boldsymbol{x}_i^n)$. If a node is originally outside the object ($\phi(\boldsymbol{x}_i^n) \geq 0$) then it should remain $\phi(\overline{\boldsymbol{x}}_i^{n+1}) \geq 0$. These cases can be combined into the constraint $\phi(\overline{\boldsymbol{x}}_i^{n+1}) \geq \min(\phi(\boldsymbol{x}_i^n), 0)$. 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 $\phi(\boldsymbol{x}_i) \geq a$ or $\phi(\boldsymbol{x}_i) = a$ can be enforced by $\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i - (\phi(\boldsymbol{x}_i) - a)\nabla\phi(\boldsymbol{x}_i)$, 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 ($\phi(\boldsymbol{x}_i^n) < 0$), then it should stay at its current depth, $\phi(\overline{\boldsymbol{x}}_i^{n+1}) = \phi(\boldsymbol{x}_i^n)$. If a node is originally outside the object ($\phi(\boldsymbol{x}_i^n) \geq 0$) 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 ($\boldsymbol{v}_i^\star \to \overline{\boldsymbol{v}}_i^{n+1}$) 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 (see Section 2.6.7 for more details).

2.6.10 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 ($\boldsymbol{v}_i^\star$) and after ($\overline{\boldsymbol{v}}_i^{n+1}$) are already available. $\Delta \boldsymbol{v}_i = \overline{\boldsymbol{v}}_i^{n+1} - \boldsymbol{v}_i^\star$ is the velocity change attributable to collisions.

In the implicit case, the collision contribution is from the last term of Equation (2.107). We compute the velocity estimate before forces as $\boldsymbol{v}_i^\star = \boldsymbol{v}_i^n + \frac{\Delta t}{m_i^n} \boldsymbol{f}_i(\boldsymbol{F}_p^{E,n+1})$. Although $\overline{\boldsymbol{v}}_i^{n+1}$ 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 $\boldsymbol{v}_i^\star$ to compute the difference for $\Delta \boldsymbol{v}_i$.

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

2.6.11 Initialization

Particle locations are initialized with Poisson disk sampling. Initial values for m_p, $\boldsymbol{x}_p^n$, and $\boldsymbol{v}_p^n = \boldsymbol{v}(\boldsymbol{x}_p^n)$ are chosen based on the needs of the example, with $\boldsymbol{v}(\boldsymbol{x})$ the desired initial velocity field. We initialize the matrix $\boldsymbol{C}_p^n = \boldsymbol{B}_p^n(\boldsymbol{D}_p^n)^{-1} = \nabla\boldsymbol{v}$ to be the gradient of the initial velocity field and $\boldsymbol{D}_p^n$ is computed from (2.92). Our initial setups have no deformation, so $\boldsymbol{F}_p^{E,n} = \boldsymbol{F}_p^{E,n+1} = \boldsymbol{I}$. We initialize our hardening parameter with $q_p^n = 0$, from which we can compute α_p^n using (2.105) and (2.106). Initial particle volume V_p^0 is computed from the seeding density.

2.7 Results

Flowing and Piling. We demonstrate the accuracy of our model by showing the characteristic behaviors of sand flowing and piling. In Figure 2.1, we simulate sand flowing inside an hourglass. The sand forms a smooth granular flow and piles up at the bottom. Figure 2.17 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. In Figure 2.21, 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 2.16, 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

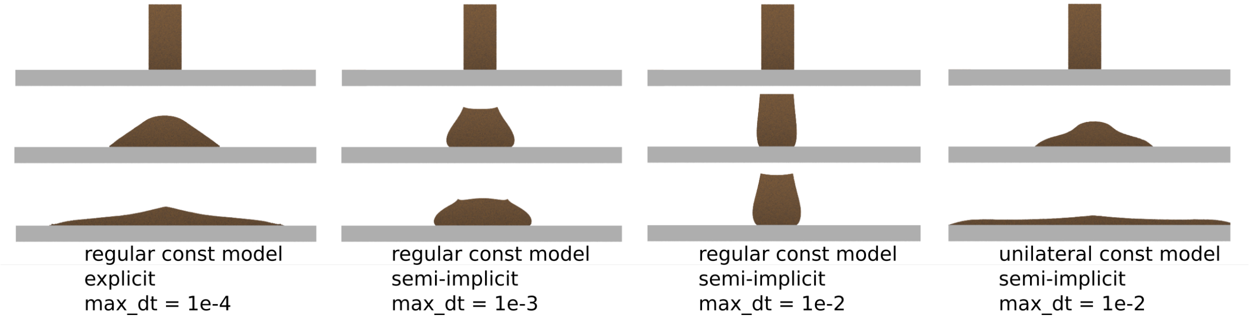

Figure 2.23: ***2D sand pile.*** *From top to bottom row: initial, middle, and final configuration of a 2D sand pile. The left column is an explicit simulation with maximum* $\Delta t = 10^{-4}$ *. The two columns in the middle are the results of semi-implicit simulation with the regular constitutive model and max* $\Delta t = 10^{-2}$ *and* 10^{-3} *respectively. Note that both suffer from artificial cohesion, although it diminishes with smaller* Δt*. On the right is semi-implicit time stepping with our unilateral energy density function with maximum* $\Delta t = 10^{-2}$*. Note that it does not suffer from artificial cohesion.*

effect is desirable.

Two-way Coupling. The benefits of using MPM include automatic self collision and coupling between different materials. In Figure 2.20, 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. In Figure 2.18, 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.

Drawing. We further demonstrate the versatility of our method by performing various tasks in a sand box. Figure 2.22 shows drawing a butterfly with a wooden stick. Figure 2.15 shows raking sand in a Zen garden.

Unilateral hyperelasticity and implicit time stepping. Figure 2.23 demonstrates the effects of our unilateral constitutive model with semi-implicit time stepping. We note that our unilateral potential removes artificial cohesion effects in the simulation of dry sand. The cohesion in this simulation is zero so the sand should not stick together. When we use the constitutive model from [KGP16] with a semi-implicit time integration scheme, we see

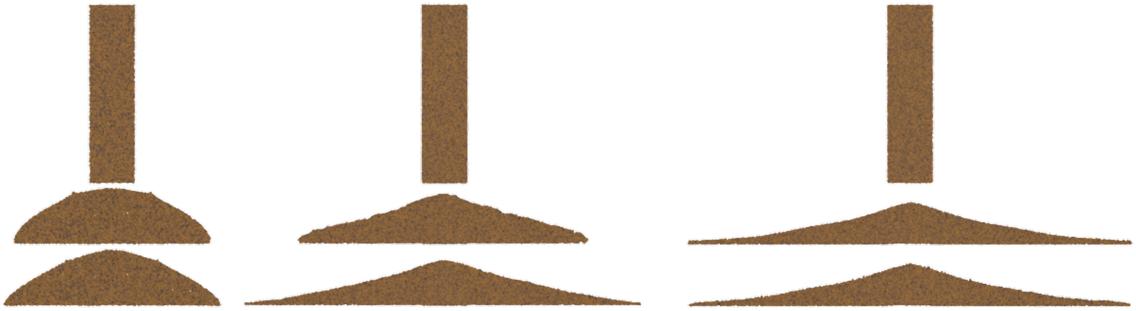

Figure 2.24: ***Fully implicit unilateral.*** *Comparison of increasingly implicit time stepping schemes for 2D sand column collapse. The simulations on the left and at the center are run with the semi-implicit scheme with the regular and unilateral energy density functions respectively. The right-most figure shows a fully implicit scheme with our modified energy density function. Note that the unilateral density function with semi-implicit time stepping yields a good approximation of the fully implicit result, while the regular energy density suffers from artificial cohesion.*

artificial cohesion that gets worse as we increase the time step size. Using our unilateral elastic energy function removes this artificial cohesion. We further demonstrate that our semi-implicit scheme gives results comparable to the more accurate, but more expensive fully implicit backward Euler scheme in Figure 2.24. The importance of choosing the right unilateral parameters is illustrated by Figure 2.8. If the regions A and B in Figure 2.7 do not closely fit the Drucker-Prager cone, then the accuracy of the simulation is compromised for large time steps.

2.8 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 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].

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.

CHAPTER 3

Wet Sand and Water

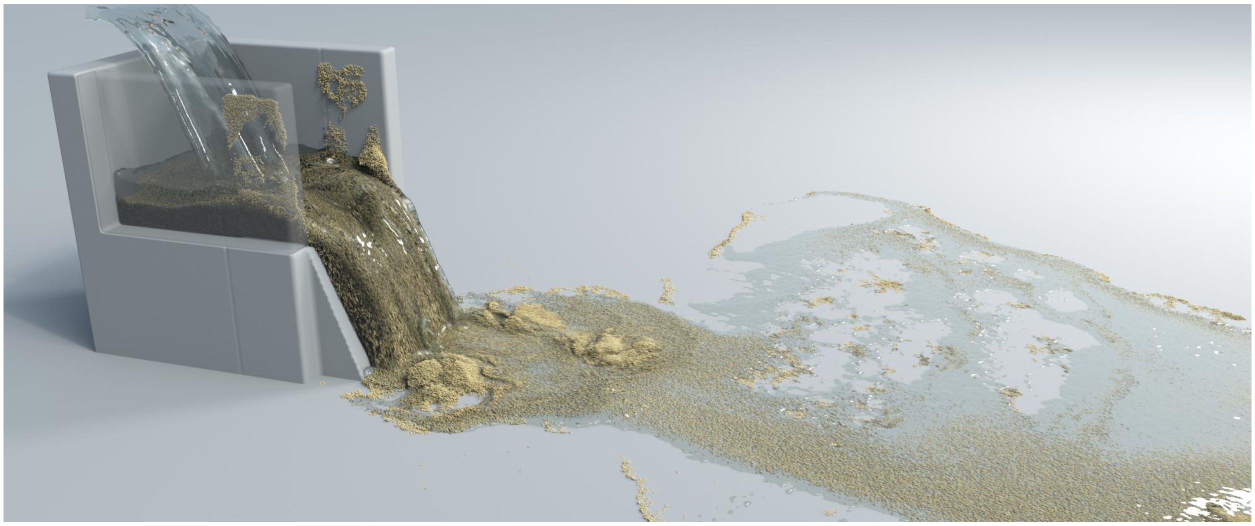

Figure 3.1: ***Dam breach****. Water pours in from a reservoir and slowly erodes a dam. As water seeps into the sand, its cohesivity decreases. When it eventually breaks, the landslide creates interesting dynamics in the debris flow.*

While wet sand is both ubiquitous and literally child's play, simulating the underlying interaction of water and sand certainly is not. This chapter presents one story on how one can simulate the phenomenon of a dam breach using Material Point Method.

3.1 Previous Work

Probably the earliest work on water and sand in computer graphics is by Peachey [Pea86]. Rungjiratananon et al. simulate sand-water interaction in real-time using a hybrid Smoothed Particles Hydrodynamics (SPH) and Discrete-Element Method (DEM) approach [RSK08]. Lenaerts and Dutre [LD09] also couple water with porous granular materials using SPH.

Notably, these approaches capture a wider range of porous phenomena than that considered in our approach. While we focus on gravity driven landslides and debris flows, their approaches more accurately capture surface tension driven effects like capillary action drawing water into dry sand. They can also handle landslides and debris flows, but they do so with SPH, whereas we develop an MPM approach that naturally allows for implicit time stepping and high resolution simulation. Other graphics approaches have shown the efficacy of hybrid Lagrangian/Eulerian approaches like FLIP and MPM, including sand [ZB05, NGL10, DB16, KGP16] and various other elastoplastic materials [SSC13, SSJ14, YSB15, RGJ15, JSS15]. Unilateral incompressibility is an effective assumption for granular materials [NGL10, AO11, IWT13, DB16].

Our approach is the first MPM technique in graphics that considers multi-species modeling for porous sand/water. However, mixture theory and multi-species simulations have been used for a wide range of effects in computer graphics. Nielsen and Osterby [NO13] simulate spray and mist with a two-continua mixture model. Takashi et al. [TFK03] use the Cubic Interpolation Propagation method to couple spray, water, and foam continua. Losasso et al. [LTK08] and Yang et al. [YLH14] also represent spray and dense water with multiple phases. Similar multi-species interaction ideas have been used for bubbles in incompressible flow [SSK05, TSS07, MMS09, RJL15]. Liu et al. [LWG08] use two continua to simulate mixtures of air and dust. Multi-species approaches have been used for miscible and immiscible fluids [BWZ10, KPN10, RLY14, HWZ15, YCR15].

Various researchers in engineering have shown the efficacy of simulating water and soil interactions with the MPM. Abe et al. [ASB14] solve coupled hydromechanical problems of fluid-saturated soil subjected to large deformation with a two grid MPM algorithm based on Biot's mixture theory. Bandara et al. use a single grid MPM method for saturated and unsaturated soils that undergo large deformations in [BFL16] and two grid MPM to represent soil skeleton and pore water layers in [BS15]. Jassim et al. [JSV13] also develop a two grid MPM approach for soil mechanics problems. Mast et al. [MAM14b] use MPM to simulate large deformation, gravity-driven landslides of porous soil. Mackenzie-Helnwein et al. [MAS10] examine the multi-species momentum exchange terms for problems with liquefaction, landslides, and sedimentation with two grid MPM.

3.2 Multiphase Continuum

We model sand and water as a multi-species continuum using mixture theory [AC76, Bor06]. With this approach, each species is given distinct material properties, and their motion is derived from distinct velocity fields. This kinematic assumption allows sand and water to occupy the same points in space at the same time to create the mixture. Since we are now dealing with multiphase, we use superscript to differentiate the differing phase quantities. We use superscript s to represent sand quantities and superscript w to represent water quantities. Whenever we use the Greek letter α as a superscript, it is understood that it can stand as s or w.

There are two material spaces $\Omega^{s,0}$ and $\Omega^{w,0}$, representing the original configuration of sand and water materials originally. The diffeomorphism $\boldsymbol{\varphi}^{\alpha}(\cdot, t)$ maps points in $\Omega^{\alpha,0}$ to points in $\Omega^{\alpha,t}$. As before, the deformation gradient and its determinant is defined as

$$\boldsymbol{F}^{\alpha}(\boldsymbol{X}, t) \coloneqq \nabla^{\boldsymbol{X}} \boldsymbol{\varphi}^{\alpha}(\boldsymbol{X}, t), \quad \text{and} \quad J^{\alpha}(\boldsymbol{X}, t) \coloneqq \det(\boldsymbol{F}^{\alpha}(\boldsymbol{X}, t)). \tag{3.1}$$

With this convention, the primary state is defined in terms of mass density $R^{\alpha}(\boldsymbol{X}, t)$ and material velocity $\boldsymbol{V}^{\alpha}(\boldsymbol{X}, t)$ in Lagrangian form. Their counterparts in Eulerian form are $\rho^{\alpha}(\boldsymbol{x}, t)$ and $\boldsymbol{v}^{\alpha}(\boldsymbol{x}, t)$, where it is understood that $\boldsymbol{x} \coloneqq \boldsymbol{\varphi}^{\alpha}(\boldsymbol{X}, t)$.

Each species obeys the following conservation of mass with respect to its own motion, i.e. in Lagrangian form

$$R^{\alpha}(\boldsymbol{X}, 0) = R^{\alpha}(\boldsymbol{X}, t) J^{\alpha}(\boldsymbol{X}, t), \quad \alpha = s, w. \tag{3.2}$$

In Eulerian form, it is written as

$$\frac{D^\alpha \rho^\alpha}{Dt}(\boldsymbol{x},t) + \rho^\alpha(\boldsymbol{x},t)\, \nabla \cdot \boldsymbol{v}^\alpha(\boldsymbol{x},t) = 0, \quad \alpha = s, w. \tag{3.3}$$

Here the material derivative operator is taken with respect to the motion of species α, and is defined as

$$\frac{D^\alpha h}{Dt}(\boldsymbol{x},t) = \frac{\partial h}{\partial t}(\boldsymbol{x},t) + \boldsymbol{v}^\alpha(\boldsymbol{x},t) \cdot \nabla h(\boldsymbol{x},t), \quad \alpha = s, w, \tag{3.4}$$

for a generic function $h : \Omega^{\alpha,t} \times (0,T) \to \mathbb{R}$. For a function $\boldsymbol{h} : \Omega^{\alpha,t} \times (0,T) \to \mathbb{R}^d$, this operator acts as

$$\frac{D^\alpha \boldsymbol{h}}{Dt}(\boldsymbol{x},t) = \frac{\partial \boldsymbol{h}}{\partial t}(\boldsymbol{x},t) + \nabla \boldsymbol{h}(\boldsymbol{x},t)\, \boldsymbol{v}^\alpha(\boldsymbol{x},t), \quad \alpha = s, w. \tag{3.5}$$

It is useful to introduce the notion of the mixture. The world space of the mixture, denoted by Ω^t is the union of the world spaces of the individual species,i.e.

$$\Omega^t = \Omega^{s,t} \bigcup \Omega^{w,t}. \tag{3.6}$$

The total mass density of the mixture can be defined as

$$\rho(\boldsymbol{x},t) := \rho^s(\boldsymbol{x},t) + \rho^w(\boldsymbol{x},t), \tag{3.7}$$

and the total momentum of the mixture as

$$\rho(\boldsymbol{x},t)\boldsymbol{v}(\boldsymbol{x},t) := \rho^s(\boldsymbol{x},t)\boldsymbol{v}^s(\boldsymbol{x},t) + \rho^w(\boldsymbol{x},t)\boldsymbol{v}^w(\boldsymbol{x},t). \tag{3.8}$$

This defines the notion of velocity for the mixture

$$\boldsymbol{v}(\boldsymbol{x},t) = \frac{\rho^s(\boldsymbol{x},t)\boldsymbol{v}^s(\boldsymbol{x},t) + \rho^w(\boldsymbol{x},t)\boldsymbol{v}^w(\boldsymbol{x},t)}{\rho(\boldsymbol{x},t)}. \tag{3.9}$$

Summing Equation (3.3) over the two phases gives the standard conservation of mass for the mixture:

$$\frac{D\rho}{Dt} + \rho \nabla \cdot \boldsymbol{v} = 0, \tag{3.10}$$

where for a generic function $h : \Omega^t \times (0, T) \to \mathbb{R}$,

$$\frac{Dh}{Dt}(\boldsymbol{x}, t) = \frac{\partial h}{\partial t}(\boldsymbol{x}, t) + \boldsymbol{v}(\boldsymbol{x}, t) \cdot \nabla h(\boldsymbol{x}, t). \tag{3.11}$$

The conservation of linear momentum for each phase is written as

$$\rho^\alpha \frac{D^\alpha \boldsymbol{v}^\alpha}{Dt} = \nabla \cdot \boldsymbol{\sigma}^\alpha + \boldsymbol{p}^\alpha + \rho^\alpha \boldsymbol{g}, \quad \alpha = s, w. \tag{3.12}$$

where $\boldsymbol{p}^\alpha$ represents the transfer of momentum due to the relative motion of the constituents, $\boldsymbol{\sigma}^\alpha$ is the partial stress tensor associated with species α, and $\boldsymbol{g}$ is the gravitational acceleration. Because $\boldsymbol{p}^\alpha$ represent the exchange of momenta between species, the sum $\sum_\alpha \boldsymbol{p}^\alpha = \boldsymbol{0}$ must be zero to not affect the total linear momentum of the mixture. Indeed with this constraint, and noting that

$$\rho \frac{Df}{Dt} = \sum_\alpha \rho^\alpha \frac{D^\alpha f}{Dt}, \tag{3.13}$$

we can show by summing over α in Equation (3.12) we get the conservation of momentum of the mixture:

$$\rho \frac{D\boldsymbol{v}}{Dt} = \nabla \cdot \boldsymbol{\sigma} + \rho \boldsymbol{g}. \tag{3.14}$$

We have introduced $\boldsymbol{\sigma} = \sum_\alpha \boldsymbol{\sigma}^\alpha$, which denotes the Cauchy stress in the mixture expressed as the sum of the partial stresses in each species. In other words, with this notion of the Cauchy stress, conservation of linear momentum for the individual species implies conservation of linear momentum for the mixture.

3.3 How to model water

We model water as fluid with no viscosity. The stress tensor for the water species can then be written as

$$\boldsymbol{\sigma}^w = p^w \boldsymbol{I}, \tag{3.15}$$

for some scalar field p^w. Furthermore, if we impose the incompressibility condition of

$$\nabla \cdot \boldsymbol{v}^w = 0, \tag{3.16}$$

then we can think of p^w as the Lagrange multiplier associated with this divergence-free constraint. As such, there are two routes we can take to model water: by solving for the pressure p^w that makes the velocity to be divergence-free, or by using an equation of states for the pressure to model water as a weakly compressible fluid. Considering the first approach, we end up with a KKT system, and it is discussed in Section 3.3.1. The approach of using weak compressibility is discussed in Section 3.3.2.

3.3.1 KKT Approach

First, we consider an energy density function of the form [SHS12, SSJ14]

$$\psi(\boldsymbol{F}^{w,E}) = \psi_\mu(\boldsymbol{F}^{w,E}) + \psi_\lambda(\boldsymbol{F}^{w,E}), \tag{3.17}$$

where

$$\psi_\mu(\boldsymbol{F}^{w,E}) := \mu \left\| \boldsymbol{F}^{w,E} - \boldsymbol{R}^{w,E} \right\|_F^2, \quad \psi_\lambda(\boldsymbol{F}^{w,E}) := \frac{\lambda}{2}(J^{w,E} - 1)^2. \tag{3.18}$$

The matrix $\boldsymbol{R}^{w,E}$ comes from the polar decomposition of $\boldsymbol{F}^{w,E} = \boldsymbol{R}^{w,E}\boldsymbol{S}^{w,E}$, with $\boldsymbol{R}^{w,E}$ being a rotation matrix and $\boldsymbol{S}^{w,E}$ is a symmetric matrix. The scalar $J^{w,E}$ is the determinant of $\boldsymbol{F}^{w,E}$. We call ψ_μ and ψ_λ to be the deviatoric and volumetric part of the energy density function respectively. It is clear that the volumetric energy density penalizes volume change.

We now revisited the arguments made by Stomakhin et al. [SSJ14] with regards to the notion of pressure associated with ψ_λ and its evolution. Note that the Cauchy stress associated by ψ_λ is given by

$$\boldsymbol{\sigma}_\lambda = \frac{1}{J^w}\left(\frac{\partial \psi_\lambda}{\partial J^{w,E}}\frac{\partial J^{w,E}}{\partial \boldsymbol{F}^{w,E}}\right)(\boldsymbol{F}^{w,E})^\intercal = -p^w \boldsymbol{I}, \tag{3.19}$$

with

$$p^w := \frac{1}{J^{w,P}}\lambda(J^{w,E} - 1), \tag{3.20}$$

where $J^{w,P} = \det(\boldsymbol{F}^{w,P})$ and $J^{w,E} = \det(\boldsymbol{F}^{w,E})$. The temporal evolution of this pressure is given by

$$\frac{Dp^w}{Dt} = -\frac{\lambda J^{w,E}}{J^{w,P}}\nabla \cdot \boldsymbol{v}^w. \tag{3.21}$$

In the absence of plasticity, we have $J^{w,E} = J^w$ and $J^{w,P} = 1$. In this case Equation (3.20) and (3.21) can be simplified to

$$p^w := \lambda(J^w - 1), \tag{3.22}$$

and

$$\frac{Dp^w}{Dt} = -\lambda J^w \nabla \cdot \boldsymbol{v}^w. \tag{3.23}$$

In this case, the momentum balance for the material is given by

$$\begin{cases} \rho^w \dfrac{D\boldsymbol{v}}{Dt} = -\nabla p^w + \rho^w \boldsymbol{g} + \nabla \cdot \boldsymbol{\sigma}^w_\mu, & \text{(3.24a)} \\ \nabla \cdot \boldsymbol{v}^w = -\dfrac{1}{\lambda J^w}\dfrac{Dp^w}{Dt}, & \text{(3.24b)} \end{cases}$$

where $\boldsymbol{\sigma}_\mu$ is the Cauchy stress associated with the energy density function ψ_μ. In the case where $\mu = 0$, Equation (3.24) is simplified to

$$\begin{cases} \rho^w \dfrac{D\boldsymbol{v}}{Dt} = -\nabla p^w + \rho^w \boldsymbol{g}, & (3.25a) \\ \nabla \cdot \boldsymbol{v}^w = -\dfrac{1}{\lambda J^w} \dfrac{Dp^w}{Dt}. & (3.25b) \end{cases}$$

When we solve Equation (3.25), we are solving for a momentum balance subject to the constraint that the divergence of the velocity field being equal to the right hand side of Equation (3.25b). When we take the limit as $\lambda \to \infty$, however, we have

$$\begin{cases} \rho^w \dfrac{D\boldsymbol{v}}{Dt} = -\nabla p^w + \rho^w \boldsymbol{g}, & (3.26a) \\ \nabla \cdot \boldsymbol{v}^w = 0, & (3.26b) \end{cases}$$

which has the standard divergence-free constraint.

For the discrete system to be solved at each time-step, we denote the discretization of the density term by a matrix $\boldsymbol{R}^w$ and the discrete divergence matrix by $\boldsymbol{D}$. The discrete gradient operator is then given by $-\boldsymbol{D}^\top$. We further denote the velocity degrees of freedom at time t^n and t^{n+1} by $\boldsymbol{v}^{w,n}$ and $\boldsymbol{v}^{w,n+1}$, the pressure degrees of freedom at step $n+1$ by $\boldsymbol{p}^{w,n+1}$, and the degrees of freedom for the interpolated $J^{w,n}$ to be $\boldsymbol{J}$. With this conventions, we may write the boundary condition constraint as

$$\boldsymbol{B}\boldsymbol{v}^{w,n+1} = \boldsymbol{b}_c, \tag{3.27}$$

for some boundary condition matrix $\boldsymbol{B}$ and vector of $\boldsymbol{b}_c$. Note that at time t^n, we have $p^{w,n} = -\lambda(J^{w,n} - 1)$. Therefore, we may approximate

$$\frac{Dp^w}{Dt} \approx \frac{p^{w,n+1} + \lambda(J-1)}{\Delta t}. \tag{3.28}$$

If $\boldsymbol{1}$ denotes the all ones vector, and $\boldsymbol{\Lambda}^{n+1}$ is the Lagrange multiplier associated with the constraint of Equation (3.27), then we may discretize Equation (3.25) as

$$\left\{\begin{aligned} \frac{\boldsymbol{R}^w\boldsymbol{v}^{w,n+1} - \boldsymbol{R}^w\boldsymbol{v}^{w,n}}{\Delta t} &= \boldsymbol{D}^\top\boldsymbol{p}^{w,n+1} - \boldsymbol{B}^\top\boldsymbol{\Lambda}^{n+1} + \boldsymbol{R}^w\boldsymbol{g}, && \text{(3.29a)}\\ \boldsymbol{D}\boldsymbol{v}^{w,n+1} &= -\frac{1}{\lambda\Delta t}\boldsymbol{J}^{-1}\left(\boldsymbol{p}^{w,n+1} + \lambda(\boldsymbol{J} - \boldsymbol{1})\right), && \text{(3.29b)}\\ \boldsymbol{B}\boldsymbol{v}^{w,n+1} &= \boldsymbol{b}_c, && \text{(3.29c)} \end{aligned}\right.$$

which can be rearranged to

$$\left\{\begin{aligned} \frac{1}{\Delta t}\boldsymbol{R}^w\boldsymbol{v}^{w,n+1} - \boldsymbol{D}^\top\boldsymbol{p}^{w,n+1} + \boldsymbol{B}^\top\boldsymbol{\Lambda}^{n+1} &= \frac{1}{\Delta t}\boldsymbol{R}^w\boldsymbol{v}^{w,n} + \boldsymbol{R}^w\boldsymbol{g}, && \text{(3.30a)}\\ -\boldsymbol{D}\boldsymbol{v}^{w,n+1} - \frac{1}{\lambda\Delta t}\boldsymbol{J}^{-1}\boldsymbol{p}^{w,n+1} &= \frac{1}{\Delta t}(\boldsymbol{1} - \boldsymbol{J}^{-1}), && \text{(3.30b)}\\ \boldsymbol{B}\boldsymbol{v}^{n+1} &= \boldsymbol{b}_c. && \text{(3.30c)} \end{aligned}\right.$$

The last equation is a KKT system and can be written as

$$\begin{pmatrix} \frac{1}{\Delta t}\boldsymbol{R}^w & -\boldsymbol{D}^\top & \boldsymbol{B}^\top \\ -\boldsymbol{D} & -\frac{1}{\lambda\Delta t}\boldsymbol{J}^{-1} & \boldsymbol{0} \\ \boldsymbol{B} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}\begin{pmatrix} \boldsymbol{v}^{w,n+1} \\ \boldsymbol{p}^{w,n+1} \\ \boldsymbol{\Lambda}^{n+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{R}^w\boldsymbol{v}^{w,n} + \boldsymbol{R}^w\boldsymbol{g} \\ \frac{1}{\Delta t}(\boldsymbol{1} - \boldsymbol{J}^{-1}) \\ \boldsymbol{b}_c \end{pmatrix}. \tag{3.31}$$

If we take the limit as $\lambda \to \infty$, we can discretize Equation (3.29) as

$$\begin{pmatrix} \frac{1}{\Delta t}\boldsymbol{R}^w & -\boldsymbol{D}^\top & \boldsymbol{B}^\top \\ -\boldsymbol{D} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{B} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}\begin{pmatrix} \boldsymbol{v}^{w,n+1} \\ \boldsymbol{p}^{w,n+1} \\ \boldsymbol{\Lambda}^{n+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{R}^w\boldsymbol{v}^{w,n} + \boldsymbol{R}^w\boldsymbol{g} \\ \boldsymbol{0} \\ \boldsymbol{b}_c \end{pmatrix}. \tag{3.32}$$

One of the main difficulties in doing this is to choose the right discretization for $\boldsymbol{D}$. We need to make sure that the discretization is stable even though it is derived from B-splines basis functions. Ideally, there will not be any spurious null-modes in the matrix. The works of Rüberg and Cirak [RC12, RC14] address this problem on a collocated grid. However, for our final examples, we settled to model water by choosing an equation of states for the water pressure and enforcing a notion of weak-compressibility. This is explained in the next section.

3.3.2 Weak compressibility

We model the water as weakly compressible [BT07] with the partial stress

$$\boldsymbol{\sigma}^w = -p^w \boldsymbol{I}, \quad p^w = k\left(\frac{1}{J^{w\gamma}} - 1\right). \tag{3.33}$$

We note that the water pressure is related to a potential ψ^w as $p^w = -\frac{\partial \psi^w}{\partial J^w}(J^w)$, with $\psi^w(J^w) = -k\left(\frac{(J^w)^{(1-\gamma)}}{1-\gamma} - J^w\right)$. This pressure p^w is designed to stiffly penalize the volume change of the water phase, which is characterized in terms of the determinant of the water deformation gradient $J^w = \det(\boldsymbol{F}^w)$. Intuitively, J^w is the ratio of the current to initial local volume of material in the water phase. It evolves as

$$\frac{D^w J^w}{Dt} = \nabla \cdot \boldsymbol{v}^w J^w. \tag{3.34}$$

Here k is the bulk modulus of the water and γ is a term that more stiffly penalizes large deviations from incompressibility.

3.4 How to model wet sand

Recall the yield condition that is used to define the behavior of dry sand in (2.28):

$$c_F \operatorname{tr}(\boldsymbol{\sigma}^s) + \|\boldsymbol{\sigma}^s - \frac{\operatorname{tr}(\boldsymbol{\sigma}^s)}{d}\boldsymbol{I}\|_F \le c_C, \tag{3.35}$$

with $d = 2, 3$ is the spatial dimension, and the parameter $c_C \ge 0$ controls the amount of cohesion in the material. A positive c_C shifts the yield surface along the hydrostatic axis, which allows the material to exhibit stress under tension and thus cohere to itself. In this multiphase work, we model cohesivity as a function of water saturation in the sand. This is naturally measured in terms of the volume fraction of water in the mixture $\phi^w = \frac{\rho^w}{\rho}$ so that

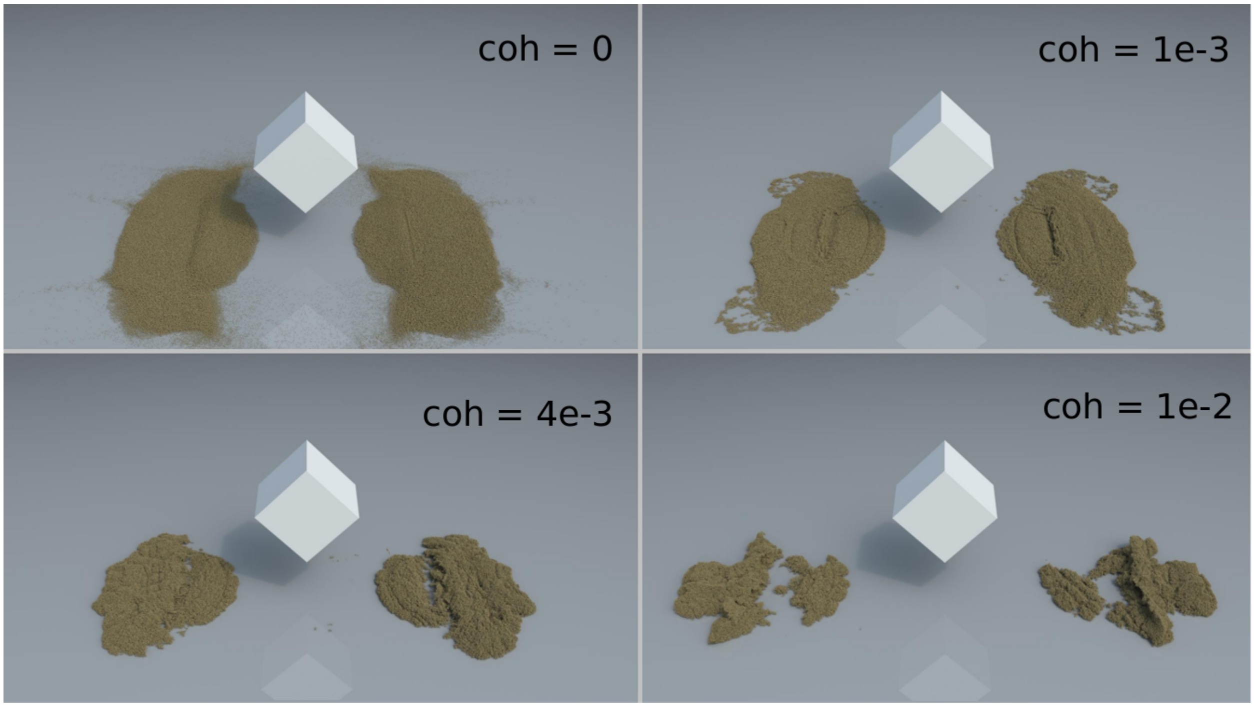

Figure 3.2: ***Dropping sand on wedges****. A box of sand with various values of cohesion is dropped onto a wedge. The top left simulation is done using an explicit time-stepping scheme, while all the others are done using a semi-implicit time-stepping scheme.*

$c_C = c_C(\phi^w)$.

3.4.1 Momentum exchange

The momentum exchange terms $\boldsymbol{p}^s, \boldsymbol{p}^w$ for water and porous sand interactions can generally be viewed as a combination of dissipative and reversible interactions [Bor06]. We follow the formulation of Bandara and Soga [BS15] because like them we are concerned primarily with gravity driven flows such as fast catastrophic landslides and debris flows. Their formulation assumes

$$\boldsymbol{p}^s = c_E\left(\boldsymbol{v}^w - \boldsymbol{v}^s\right) + p^w \nabla \phi^w, \;\; \boldsymbol{p}^w = -\boldsymbol{p}^s \tag{3.36}$$

where $c_E = \frac{n^2 \rho^w g}{\hat{k}}$ and n is the sand porosity, $\hat{k}$ is the sand permeability and g is the gravitational acceleration, $\phi^w = \frac{\rho^w}{\rho}$ is the water volume fraction and p^w is the water pressure. The first term represents viscous forces generated by sand particles moving through the fluid. Although it can be conceived from the view of an idealized particle moving through a Stokes fluid, it simply amounts to a Coulomb-friction-like response [MAS10]. The second term is often called the "buoyancy term" in mixture theories [RS13]. It can be conceived from entropy equilibrium constraints or from the physical consideration that the pore fluid pressure multiplying the porosity is appropriate notion of reversible pressure, however there is some debate about its appropriateness outside of immiscible mixtures [Dru00]. Mackenzie-Helnwein et al. [MAS10] omit the second term and view the momentum exchange terms as purely dissipative processes. We also follow this approach for the majority of the examples presented in the paper, however we include one example demonstrating its effect in Figure 3.4. Even with the inclusion of the active term from [BFL16], we are only capable of simulating a rather narrow range of porous media phenomena. While this is sufficient for landslides and debris flows, phenomena such as capillary action drawing water into dry sand is not achievable in our approach.

3.4.2 Cohesion and Saturation

As in Robert and Soga in [RS13], we assume that the sand cohesion varies as a function of water saturation. One option to measure saturation is as the percentage of water in the mixture which we estimate as the ratio of the density of the water phase to the total density $\phi^w = \frac{\rho^w}{\rho}$. We can also choose a heuristic method to model saturation based on an indicator function tracking the overlap region between water and sand. This is explained in Section 3.5.3.3.

The cohesion of sand is zero when it is completely dry ($\phi^w = 0$). Intuitively, as dry sand becomes saturated with more water, the cohesivity of the wet sand should increase as wet sand tends to better hold its shape. Indeed this was observed in the work of Robert and Soga in [RS13]. However, they also observe that this increase only continues to a maximal

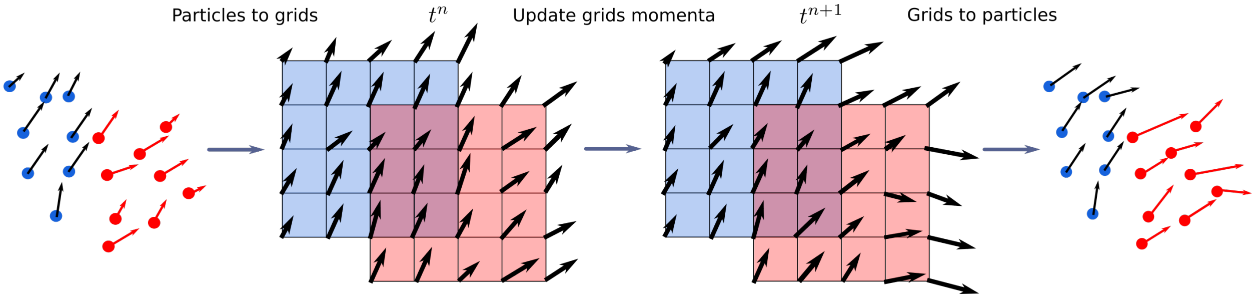

Figure 3.3: ***MPM algorithm with two grids****. We use separate water and sand grids. The blue and red dots denote water and sand particles respectively. In the overlapping region, we compute the momentum exchange term between the two species. We also compute the force based on individual constitutive model.*

value $c_C^{\max}$ when the saturation is around $\phi^w = 0.4$. Beyond this point the sand becomes more compliant to flowing and less cohesively elastic. In all of our multispecies examples, we model water interaction with wet sand that is capable of holding its shape. We thus set the sand cohesion to be initially maximal, even in the absence of the water phase. Based on the observations in Robert and Soga [RS13], we then assume that the cohesion decreases linearly with increasing saturation beyond this point (with cohesion equal to zero at full saturation $\phi^w = 1$).

3.5 Discretization

The overall MPM algorithm in this case is very similar to the one described in the first chapter. As in other approaches in the engineering literature [BS15, JSV13, MAS10], we use two sets of grids (Figure 3.3): one for the solid particles and the other for the water particles. This is the primary difference between a multispecies discretization and a single species MPM algorithm.

First, we explain the notation used in the discretization section, some of which have been used in Section 3.3.1. There are two sets of grids: one is associated with sand material and the other is associated with water. As with the continuous equations, the superscript $\alpha = s, w$ indicates the corresponding species. Whenever a symbol has a subscript i or j, this denotes

Variable	Where	Species	Meaning
$\boldsymbol{g}$	-	-	gravitational constant
c_E	-	-	drag coefficient
m_p^α	particles	both	particle mass
$V_p^{\alpha 0}$	particles	both	initial particle volume
$\boldsymbol{x}_p^{\alpha,n}$, $\boldsymbol{x}_p^{\alpha,n+1}$	particles	both	particle position
$\boldsymbol{v}_p^{\alpha,n}$, $\boldsymbol{v}_p^{\alpha,n+1}$	particles	both	particle velocity
$\boldsymbol{F}_p^n$, $\boldsymbol{F}_p^{n+1}$	particles	matrix	deformation gradient
$\boldsymbol{F}_p^{sE,n}$, $\boldsymbol{F}_p^{sE,n+1}$	particles	sand	sand elastic deformation gradient
$\boldsymbol{F}_p^{sP,n}$, $\boldsymbol{F}_p^{sP,n+1}$	particles	sand	plastic deformation gradient
$J_p^{w,n}$, $J_p^{w,n+1}$	particles	water	determinant deformation gradient
s_p^s	particle	sand	water saturation
v_{cp}^s	particle	sand	volume correction scalar
$(\nabla \boldsymbol{v})_p^\alpha$	particles	matrix	grid-based velocity gradient
$m_i^{\alpha,n}$	grid	both	grid node mass
$\boldsymbol{v}_i^{\alpha,n}$	grid	both	rasterized velocity
$\boldsymbol{v}_i^{\alpha,n+1}$	grid	both	final grid velocity
$\boldsymbol{x}_i^{\alpha,n}$	grid	both	Cartesian grid node locations
$\hat{\boldsymbol{x}}_i^{\alpha,n+1}$	grid	both	grid positions moved by $\boldsymbol{v}_i^{\alpha,n+1}$
$\phi_i^{s,n+1}$	grid	mixed	water saturation
$\boldsymbol{f}_i^\alpha$	grid	both	internal forces

Table 3.1: ***Table of notation for multiphase simulation.***

one degree of freedom in grid node index i or j. Subscript p denotes attributes that belong to a particle. A symbol that is not followed by a subscript refers to the whole collection of grid nodes as degrees of freedom. For example $\boldsymbol{v}^{s,n+1}$ refers to all of the sand grid nodes that are active at step $n+1$.

The overview of the algorithm is as follows:

1. **Transfer to grids**: Transfer the mass and momentum of each species to its corresponding grid (§3.5.1).

2. **Update grids momenta**: Solving for the coupled water and sand grid velocities using semi-implicit backward Euler (§3.5.2).

3. **Update particles**: Update all particle state, including the cohesion based on saturation as well as plasticity return mappings (§3.5.3).

Although many of these steps are very similar to the steps described in Chapter 2 (with the modification of having superscript α in some of the symbols), we include them here for completeness.

3.5.1 Transfer to grid

We transfer mass and momentum from sand and water particles to their respective grid just like we did in the first chapter. For each species α, the particle p interacts with the grid node ${}^{\alpha}_{i}$ with weight $w_{ip}^{\alpha,n} = N(\boldsymbol{x}_p^{\alpha,n} - \boldsymbol{x}_i^{\alpha})$. The weight is computed using the quadratic B-spline interpolation kernel. Just as in Equation (2.91) mass on the grid is computed according to

$$m_i^{\alpha,n} = \sum_p w_{ip}^{\alpha,n} m_p^{\alpha}, \tag{3.37}$$

and velocity according to

$$\boldsymbol{v}_i^{\alpha,n} = \frac{1}{m_i^{\alpha,n}} \sum_p w_{ip}^{\alpha,n} m_p^{\alpha} \left(\boldsymbol{v}_p^{\alpha,n} + \boldsymbol{C}_p^{\alpha,n}\left(\boldsymbol{x}_i^{\alpha,n} - \boldsymbol{x}_p^{\alpha,n}\right)\right), \tag{3.38}$$

where the matrix $\boldsymbol{C}_p^{\alpha,n}$ is an extra matrix stored per particle which defines an affine velocity field local to the particle [JSS15]. $\boldsymbol{C}_p^{\alpha,n}$ is initialized with $\boldsymbol{C}_p^{\alpha,0} = \boldsymbol{0}$ and updated at the end of the previous time step during the grid-to-particle transfer (Equation (3.50)).

3.5.2 Update Grids Momenta

Using MPM, the forces in the sand and water phases are computed as

$$\boldsymbol{f}_i^s(\hat{\boldsymbol{x}}^s) = -\frac{\partial \psi^s}{\partial \hat{\boldsymbol{x}}_i^s} = -\sum_p V_p^0 \left(\frac{\partial \psi^U}{\partial \boldsymbol{F}^s}(\boldsymbol{F}_p^{sE}(\hat{\boldsymbol{x}}^s))\right)(\boldsymbol{F}_p^{sE,n})^{\top} \nabla w_{ip}^{s,n}, \tag{3.39}$$

$$\boldsymbol{f}_i^w(\hat{\boldsymbol{x}}^w) = -\frac{\partial \psi^w}{\partial \hat{\boldsymbol{x}}_i^w} = -\sum_p V_p^0 \left(\frac{\partial \psi^w}{\partial J^w}(J^w(\hat{\boldsymbol{x}}^w))\right) J^{w,n} \nabla w_{ip}^{w,n}. \tag{3.40}$$

As in [SSC13] we think of $\hat{\boldsymbol{x}}_i^{\alpha}$ as the position of the grid node i corresponding to species α that has been deformed from its original position $\boldsymbol{x}_i^{\alpha}$ by an amount of $\Delta t\, \boldsymbol{v}_i^{\alpha,n+1}$, i.e. $\hat{\boldsymbol{x}}_i^{\alpha} = \hat{\boldsymbol{x}}_i^{\alpha}(\boldsymbol{v}_i^{\alpha,n+1}) = \boldsymbol{x}_i^{\alpha,n} + \Delta t\, \boldsymbol{v}_i^{\alpha,n+1}$. The discrete momentum balance to be solved is

$$m_i^{s,n}\left(\boldsymbol{v}_i^{s,n+1} - \boldsymbol{v}_i^{s,n}\right) = \Delta t\, \left(\boldsymbol{f}_i^s\left(\hat{\boldsymbol{x}}^s\right) + m_i^{s,n}\boldsymbol{g} + \boldsymbol{d}_i^s(\hat{\boldsymbol{x}})\right) \tag{3.41}$$

$$m_j^{w,n}\left(\boldsymbol{v}_j^{w,n+1} - \boldsymbol{v}_j^{w,n}\right) = \Delta t\, \left(\boldsymbol{f}_j^s\left(\hat{\boldsymbol{x}}^w\right) + m_j^{w,n}\boldsymbol{g} + \boldsymbol{d}_j^w(\hat{\boldsymbol{x}})\right), \tag{3.42}$$

where the discrete interaction term is given by

$$\boldsymbol{d}^s_{ij}(\hat{\boldsymbol{x}}) = -c_E\, m^s_i\, m^w_j (\boldsymbol{v}^{s,n+1}_i - \boldsymbol{v}^{w,n+1}_j), \tag{3.43}$$

$$\boldsymbol{d}^w_{ji}(\hat{\boldsymbol{x}}) = c_E\, m^s_i\, m^w_j (\boldsymbol{v}^{s,n+1}_i - \boldsymbol{v}^{w,n+1}_j), \tag{3.44}$$

for some drag coefficient c_E. Setting

$$\boldsymbol{M} = \begin{pmatrix} \boldsymbol{M}^{s,n} & \\ & \boldsymbol{M}^{w,n} \end{pmatrix}, \ \boldsymbol{v} = \begin{pmatrix} \boldsymbol{v}^s \\ \boldsymbol{v}^w \end{pmatrix}, \ \boldsymbol{f}(\hat{\boldsymbol{x}}(\boldsymbol{v}^{n+1})) = \begin{pmatrix} \boldsymbol{f}^s\,(\hat{\boldsymbol{x}}^s) \\ \boldsymbol{f}^w\,(\hat{\boldsymbol{x}}^w) \end{pmatrix},$$

and $\boldsymbol{D}^i$ to be the drag coefficient matrix derived from Equations (3.43) and (3.44), we can write the coupled system as

$$\left(\boldsymbol{M} + \Delta t\, \boldsymbol{D}^i\right) \boldsymbol{v}^{n+1} = \boldsymbol{M}\boldsymbol{v}^n + \Delta t \left(\boldsymbol{M}\boldsymbol{g} + \boldsymbol{f}(\hat{\boldsymbol{x}}(\boldsymbol{v}^{n+1}))\right). \tag{3.45}$$

At each time step, we solve this nonlinear system using a few iterations of a modified Newton's method. Since the matrix $\boldsymbol{D}^i$ is symmetric, and both $\boldsymbol{f}^s$ and $\boldsymbol{f}^w$ are derived as the negative gradient of a potential, the whole system is symmetric when linearized (assuming the effects of plasticity are ignored in the linearization) and can be solved using MINRES. We note that we do not include the effect of plasticity when computing the derivatives of $\boldsymbol{f}^s$. Doing otherwise results in non-symmetric sand force derivatives which would require GMRES. Our omission of these terms in the linearization of the system is a modification to the standard Newton's method. However, it is essential that we use implicit time stepping because of the stiff momentum exchange terms and our lagged plasticity approach is the key to making this efficient.

3.5.3 Update Particles

3.5.3.1 Update J^w.

We do not keep track of the deformation gradient $\boldsymbol{F}^w$ of water particles, instead we keep track of its determinant J^w, which is updated according to the discretization of Equation (3.34), i.e.

$$J_p^{w,n+1} = \left(\boldsymbol{I} + \Delta t \ \operatorname{tr}(\nabla \boldsymbol{v}_p^{w,n+1})\right) J^{w,n}.$$

We found that in practice, this discretization tends to offer more stability than the alternative of evolving $\boldsymbol{F}^w$ followed by computing its determinant.

3.5.3.2 Update $\boldsymbol{F}^s$.

Because we ignore the effects of plasticity during the implicit solve for momenta, $\hat{\boldsymbol{F}}^{sE,n+1}$ evolves with the grid during the grid momentum update as in Equation (2.101):

$$\hat{\boldsymbol{F}}_p^{sE,n+1} = \left(\boldsymbol{I} + \Delta t \ \nabla \boldsymbol{v}_p^{s,n+1}\right) \boldsymbol{F}^{sE,n}. \tag{3.46}$$

$\hat{\boldsymbol{F}}_p^{sE,n+1}$ is later processed for plasticity at the end of the time step to define $\boldsymbol{F}_p^{sE,n+1}$, which we discuss below.

3.5.3.3 Saturation based cohesion

We define the water saturation on sand particles based on a heuristic. First, we populate a grid whose domain is the union of the sand grid and water grid domain. We mark each node that has a non-zero mass for both sand and water species as $\phi_i^{w,n+1} = 1$, otherwise $\phi_i^{w,n+1} = 0$. One can think of this grid as tracking an indicator function of the overlap region between the sand and water constituents. We then compute the saturation on the sand particle by interpolating from the grid to the sand particle according to

$$\phi_p^{s,n+1} = \sum_i w_{s,ip}^n \phi_i^{w,n+1}. \tag{3.47}$$

This approximates the saturation as equal to one (maximal) deep in the interior of the overlap region with a ramp to zero exterior to the region. In all of our multi-species examples, we assume that the wet sand is already saturated and has reached its maximum cohesion level. Hence, any amount of additional water saturation will lower the cohesion level of the sand. We vary cohesion in a linear fashion as a function of water saturation as discussed in Section 3.4.2

$$c_{Cp}^{s,n+1} = c_{Cp}^{s,0}(1 - \phi_p^{w,n+1}) \tag{3.48}$$

where we note that the saturation is always in $(0,1)$. The approximation of the saturation in Equation (3.47) has errors biased towards full saturation in the interior. This naturally leads to more rapid failure in the landslides and debris flows we consider in our examples since the cohesion decreases more rapidly than it should. This is an extreme simplification to correct behavior defined in Robert and Soga [RS13], but we found that it was effective for simulating these phenomena.

3.5.3.4 Update position and velocity.

Velocity is updated according to

$$\boldsymbol{v}_p^{\alpha,n+1} = \sum_i w_{ip}^{\alpha,n} \boldsymbol{v}_i^{\alpha,n+1}. \tag{3.49}$$

Since we use quadratic B-splines in all of our multiphase examples, the velocity spatial derivative matrix $\boldsymbol{C}_p^\alpha$ is updated according to

$$\boldsymbol{C}_p^{\alpha,n+1} = \sum_i w_{ip}^n \boldsymbol{v}_i^{\alpha,n+1} \left(\left(\frac{4}{dx^2} \right) (\boldsymbol{x}_i^{\alpha,n} - \boldsymbol{x}_p^{\alpha,n}) \right)^\top, \tag{3.50}$$

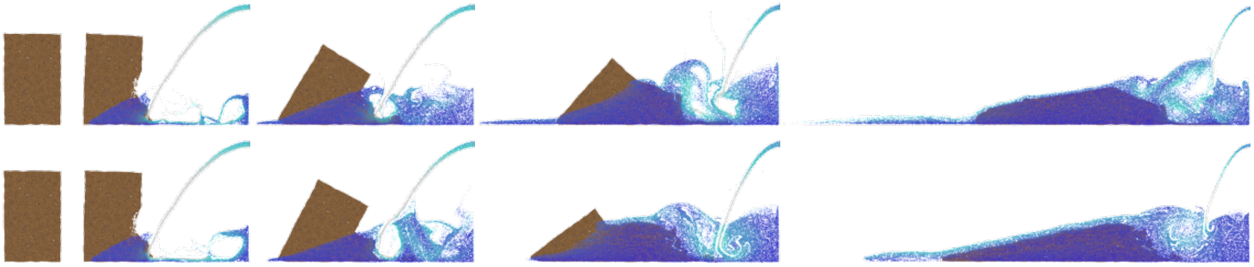

Figure 3.4: ***2D dam breach.*** *A 2D dam breach simulation showing the effect of reversible term from Bandara and Soga [BS15]. The figure depicts the beginning, middle, and the end of a dam breach scenario. At the top is a simulation with no reversible term (as in [MAS10]) and at the bottom is a simulation with the reversible term.*

here dx is the Eulerian grid spacing. Lastly position is updated according to

$$\boldsymbol{x}_p^{\alpha,n+1} = \boldsymbol{x}_p^{\alpha,n} + \Delta t\, \boldsymbol{v}_p^{\alpha,n+1}. \tag{3.51}$$

3.6 Implementation and Results

We use the sparse grid structure provided by OpenVDB [Mus13].

In Figure 3.2 we demonstrate how varying cohesion gives rise to different wet sand behaviors. In Figure 3.5 we demonstrate our approach with an example that is representative of the types of gravity driven flows we are interested in with our approach. As water flows into the wall of a dam, the saturation increases weakening it. The cohesion of sand decreases with saturation and the dam eventually breaks.

We demonstrate the effect of the active component of the momentum exchange terms in Equation (3.36) using a simulation of a 2D dam breach, shown in Figure 3.4. Again water pours in from a reservoir and slowly erodes a retaining wall. We note that the active term has only a subtle effect on the bulk dynamics of the motion for these types of flows. We

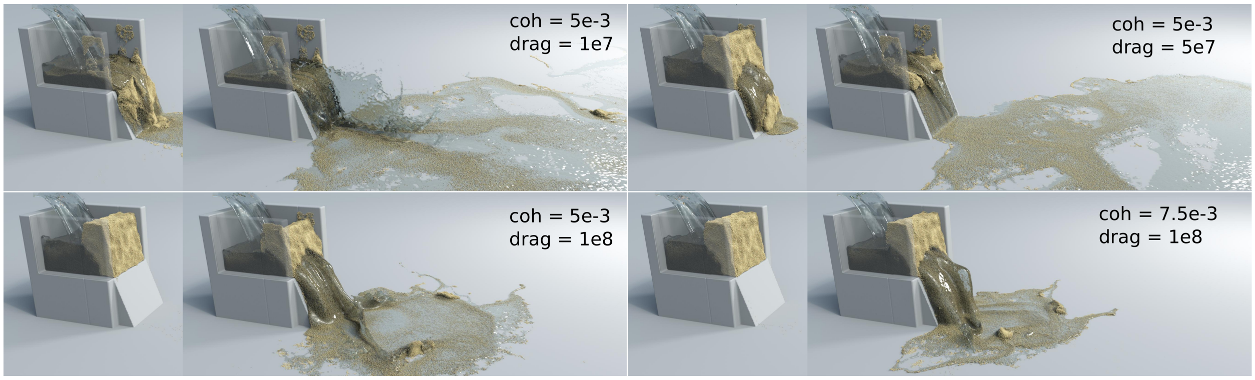

Figure 3.5: ***When the levee breaks.*** *Here we demonstrate the effects of our parameters on levee wall integrity. A simulation with lower drag momentum exchange coefficient and lower cohesion will fail more easily.*

discretize the active term by adding

$$\sum -p_p^{w,n} \nabla w_{ip}^{s,n} \frac{m_i^{w,n}}{m_i^{s,n} + m_i^{w,n}} \tag{3.52}$$

to the water drag term in Equation (3.44). We then define the solid drag term to be equal and opposite to the water drag in accordance with the zero-net-sum nature of the momentum exchange.

3.7 Limitations and future work

Our approach has a number of limitations. The momentum exchange model we use in the water/sand multispecies examples is rather simplified. While adequate for gravity driven flows like landslides and levee breaches, it is inadequate for capillary driven phenomena like water being drawn in to dry sand. Such phenomena has been captured by prior approaches like that of Lenaerts and Dutre [LD09]. Furthermore we fail to capture behavior like those in Rungjiratananon et al. [RSK08] where surface tension effects in wetting are more accurately captured.

Although our approximation to the dependence of sand cohesion on saturation is useful for facilitating rapid failure of water/sand mixtures, it is an extreme simplification to the correct behavior defined in Robert and Soga [RS13] and this compromises its accuracy dramatically. This reduces the applicability of our approach outside of visually plausible simulation applications.

Large values of the momentum exchange coefficient c_E can lead to ill-conditioning in the linear systems that arise during implicit time stepping. We found that these cases required many MINRES iterations to resolve and lead to excessive run times. This complicated the simulation of slurry materials where the water and sand remain mixed. In the future we would like to examine appropriate preconditioners to improve the performance. Also, while we omit or use a very simplistic buoyancy term for the reversible momenta exchange in the $\boldsymbol{p}^\alpha$ equations, we would like to examine the addition of more accurate terms to produce phenomena like absorbent sponges interacting with liquids. Lastly, we would like to examine the suitability of our multiple grid MPM framework for the simulation of more general multi-species interactions like chemically reacting flow.

CHAPTER 4

Redistancing

4.1 Introduction

The level set method, as first introduced by Osher and Sethian in [OS88] has been proven to be a really useful and effective computational frameworks in areas such as computational fluid dynamics, seismic simulation, minimal surface computations, image and geometric processing, among many other things. The *signed distance function* has proven to be a very useful concept within this framework.

Given a set $\Omega \subset \mathbb{R}^d$ ($d = 2$ or 3) and its boundary $\partial\Omega$, the distance function $\hat{d}(\boldsymbol{x})$ measures the shortest distance between the set $\partial\Omega$ with the point $\boldsymbol{x}$. The signed-distance function is defined as

$$\hat{\phi}(\boldsymbol{x}) := \begin{cases} -\hat{d}(\boldsymbol{x}) & \boldsymbol{x} \in \Omega \setminus \partial\Omega, \\ 0 & \boldsymbol{x} \in \partial\Omega, \\ \hat{d}(\boldsymbol{x}) & \boldsymbol{x} \notin \Omega \cup \partial\Omega. \end{cases}$$

However, when one advects a signed distance function over time, the result of this advection is not a signed distance function. As such one often needs to do *redistancing.* We revisit this problem by proposing an algorithm that is highly parallelizable.

One way to formalize the problem of compute $\hat{\phi}(\boldsymbol{x})$ is by solving an Eikonal equation.

So for a given set Ω, let $\phi^0 : \mathbb{R}^d \to \mathbb{R}$ be a function with the property

$$\begin{cases} \phi^0(\boldsymbol{x}) < 0, & \boldsymbol{x} \in \Omega \setminus \partial\Omega, \quad (4.1a) \\ \phi^0(\boldsymbol{x}) = 0, & \boldsymbol{x} \in \partial\Omega, \quad (4.1b) \\ \phi^0(\boldsymbol{x}) > 0, & \boldsymbol{x} \notin \Omega \cup \partial\Omega. \quad (4.1c) \end{cases}$$

Finding the signed distance function $\hat{\phi}(\cdot)$ is equivalent as finding the solution to

$$\begin{cases} \|\nabla^{\boldsymbol{x}} \phi(\boldsymbol{x})\|_2 = 1, & (4.2a) \\ \operatorname{sgn}(\phi(\boldsymbol{x})) = \operatorname{sgn}(\phi^0(\boldsymbol{x})), & (4.2b) \end{cases}$$

where $\operatorname{sgn}(x)$ is the sign function, and $\|\cdot\|$ is the ℓ_2-norm.

It is good to point out, as noted in the introduction of this thesis, that the algorithm that we propose here can also be modified to handle a more general Hamilton-Jacobi equation. It is conceivable that this algorithm can prove to be useful in plasticity theory.

4.2 Previous works

There is extensive previous work related to the solution of (4.2). The most commonly used methods are the fast marching method (FMM) [Tsi95, Set96] and the fast sweeping method (FSM) [Zha05]. First proposed by Tsitsiklis [Tsi95] using optimal control, the fast marching method was independently developed by Sethian in [Set96] based on upwind difference schemes. It is similar to Dijkstra's method [Dij59] for finding the shortest path between nodes in a graph. The fast marching method uses upwind difference stencils to create a discrete data propagation consistent with the characteristics of the eikonal equation. Sorting is used to determine a non-iterative update order that minimizes the number of times that a point is utilized to create a strictly increasing (or decreasing) propagation. The operation count is $\mathcal{O}(N \log(N))$ where N is the number of grid points and the $\log(N)$ term is a consequence of the sorting. Fast sweeping is similar, but it uses a simpler propagation. Rather than using the optimal update ordering that requires a heap sort, a Gauss-Seidel iterative approach is used with alternating sweep directions. Typically, grid axis dimensions are used as the sweep directions. For $\mathbb{R}^d$ it only requires 2^d propagation sweeps of updating to properly evaluate every point.

Notably, both the FMM and FSM approaches create data flow dependencies since information is propagated from the zero isocontour outwards and this complicates parallel implementation. Despite this, various approaches have achieved excellent performance with parallelization. The Gauss-Seidel nature of FSM makes it more amenable to parallelization than FMM. Zhao initially demonstrated this in [Zha07] where each sweep direction was assigned to an individual thread with the final updated nodal value being the minimum nodal value from each of the threads. This method only allowed for a low number of threads and further scaling was achieved by splitting the individual sweeps into subdomain sweeps with a domain decomposition approach. However, this strategy can require more sweep iterations than the original serial FSM and the required iterations increase with the number of domains which reduces parallel efficiency. Detrixhe et al. [DGM13] developed a parallel FSM that scales in an arbitrary number of threads without requiring more iterations than in the serial case. Rather than performing grid-axis-aligned Gauss-Seidel sweeps, they use Cuthill-McKee ordering (grid-diagonal) to decouple the data dependency. Since the upwind difference stencil only uses grid axis neighbors, nodes along a diagonal do not participate in the same equation and can thus be updated in parallel trivially. They extended these ideas to hybrid distributed/shared memory platforms in [DG16]. They use a domain decomposition strategy similar to Zhao [Zha07] to divide the grid among available compute nodes and a fine grained shared memory method within each subdomain that utilizes their approach in [DGM13] to achieve orders of magnitude performance increases.

FMM is more difficult to implement in parallel, however even Tsitsiklis [Tsi95] developed a parallel FMM algorithm using a bucket data structure. A number of approaches use domain decomposition ideas similar to Zhao [Zha07] and Detrixhe et al. [DG16] to develop parallel FMM [Her03, YS17, JW07, BCG11]. In these approaches the grid is typically divided into disjoint sub grids with a layer of ghost nodes continuing into adjacent neighbors. Each sub grid is updated in parallel with the FMM update list typically leading to rather elaborate communication between threads. Jeong et al. [JW07] developed the fast iterative method (FIM), which is a parallel approach using domain decomposition but with a looser causal relationship in the node update list to localize updates for Single Instruction Multiple Data (SIMD) level parallelism. Simplifications to the update list in FMM improve parallel scaling, but tend to increase the number of worst case iterations. Dang et al. [DE14] extended FIM to a coarse/fine-grained approach based on domain decomposition with load balancing via master/worker model that allowed for efficient performance on heterogeneous platforms.

Recently an interesting alternative to FMM and FSM has been proposed. Darbon and Osher [DO16] and Lee et al. [LDO17] utilize the Hopf-Lax formulation of the solution to the Hamilton-Jacobi form of the eikonal equation. Notably, the signed distance at an arbitrary point is obtained without the need of distance information from neighboring points. This allows for the solution at any given point in any order and prevents the need for communication across cores which greatly simplifies parallel implementation. Furthermore, this inherently allows for updates done only in a narrow band near the zero-isocontour. FSM must solve over the entire domain, and while FMM can be done in a narrow band, FMM methods are generally more difficult to implement in parallel. These aspects make the Hopf-Lax approaches in [LDO17, DO16] very compelling for parallel architectures. In this paper, we extend the work of Lee et al. to handle functions defined via interpolation over a regular grid. Lee et al. demonstrated compelling results with abstractly defined functions. However, treatment of grid-based functions is essential for practical application in level set methods. We demonstrate the effectiveness of our approach with Graphics Processing Unit (GPU) parallel implementation.

4.3 Our approach in continuous setting

In light of the property of $\phi^0(\cdot)$ defined by (4.1), we consider a function $\phi(\boldsymbol{x},t)$ with $\phi(\boldsymbol{x},0) = \phi^0(\boldsymbol{x})$. The equation for the isocontour of this function is

$$\phi(\boldsymbol{x},t) = c, \tag{4.3}$$

for some constant c. It is also useful to define the set Γ_t to be the zero isocontour of the function $\phi(\boldsymbol{x},t)$, i.e.

$$\Gamma_t := \{\boldsymbol{x} \in \mathbb{R}^d : \phi(\boldsymbol{x},t) = 0\}. \tag{4.4}$$

Note that with $t = 0$, the set Γ_0 is $\partial\Omega$.

Taking the material derivative of (4.3) gives

$$\frac{D\phi}{Dt}(\boldsymbol{x},t) = \frac{\partial\phi}{\partial t}(\boldsymbol{x},t) + \frac{\partial\boldsymbol{x}}{\partial t}(\boldsymbol{x},t) \cdot \nabla\phi(\boldsymbol{x},t) = 0, \tag{4.5}$$

where $\dfrac{\partial\boldsymbol{x}}{\partial t}(\boldsymbol{x},t)$ is the advection velocity of the level set. In the case where

$$\frac{\partial\boldsymbol{x}}{\partial t}(\boldsymbol{x},t) := a\frac{\nabla\phi(\boldsymbol{x},t)}{\|\nabla\phi(\boldsymbol{x},t)\|_2}, \tag{4.6}$$

then (4.5) describes the advection of the level set of $\phi(\boldsymbol{x},t)$ with speed a in a direction that is normal to the isocontour of the function. When the constant a is positive the front propagates in an outward normal direction. When $a < 0$, then it moves in the inward normal direction. When a is zero, the isocontour does not move over time. In particular, when $a = 1$, the set Γ_t then defines an isocontour of a set which is a distance t from $\partial\Omega$.

We are then led to consider the initial value problem

$$\begin{cases} \dfrac{\partial \phi}{\partial t}(\boldsymbol{x}, t) = -\|\nabla \phi(\boldsymbol{x}, t)\|_2, & \text{(4.7a)} \\ \phi(\boldsymbol{x}, 0) = \phi^0(\boldsymbol{x}). & \text{(4.7b)} \end{cases}$$

This reasoning gives us a strategy to compute the signed distance function $\hat{\phi}(\boldsymbol{x})$ from a boundary $\partial\Omega$. Given $\boldsymbol{x} \in \mathbb{R}^d$, we compute $\phi(\boldsymbol{x}, t)$ and solve for the time $\hat{t}$ such that $\phi(\boldsymbol{x}, \hat{t}) = 0$. The time $\hat{t}$ will be the distance from the point $\boldsymbol{x}$ to the boundary $\partial\Omega$. The sign of $\hat{\phi}(\boldsymbol{x})$ is determined by the sign of $\phi^0(\boldsymbol{x})$. This describes the big picture of the algorithm.

Note that for a given point $\boldsymbol{x}$, this computation is fully parallelizable. There are two details that need to be spelled out, namely the choice of algorithm to find the root of $\phi(\boldsymbol{x}, \cdot) = 0$ and a way to compute the *actual value* of $\phi(\boldsymbol{x}, t)$ for a given pair $(\boldsymbol{x}, t)$. The first problem is solved discretely using modified secant method, and is explained in Section 4.5.1. Computing $\phi(\boldsymbol{x}, t)$ is done by using Hopf-Lax formula and is discussed in the next section.

4.4 The Hopf-Lax Formula

In this chapter, we discuss the Hopf-Lax formula in the context of a more general Hamilton-Jacobi equation.

A Hamilton-Jacobi equation is an initial value problem of the form

$$\begin{cases} \dfrac{\partial u}{\partial t}(\boldsymbol{x}, t) + H\left(\nabla u(\boldsymbol{x}, t)\right) = 0, & \text{(4.8a)} \\ u(\boldsymbol{x}, 0) = g(\boldsymbol{x}). & \text{(4.8b)} \end{cases}$$

The function H is usually assumed to be convex, and the function g is assumed to be Lipschitz continuous. It is obvious that (4.7) is a Hamilton-Jacobi equation with $H(\cdot) = \|\cdot\|_2$.

The Hopf-Lax formula gives the solution to the Hamilton-Jacobi equation, and is defined in terms of the Legendre transform of H.

Definition 4.4.1. *Given a convex function $H : \mathbb{R}^d \to \mathbb{R}$. The Legendre transform $H^* : \mathbb{R}^d \to \mathbb{R}$ of H is defined as*

$$H^*(\boldsymbol{x}) := \sup_{\boldsymbol{y} \in \mathbb{R}^d} \{\boldsymbol{x} \cdot \boldsymbol{y} - H(\boldsymbol{y})\}. \tag{4.9}$$

For H being the ℓ_2-norm, its Legendre transform is

$$H^*(\boldsymbol{x}) := \begin{cases} 0 & \|\boldsymbol{x}\|_2 \leq 1, \\ +\infty & \text{otherwise.} \end{cases} \tag{4.10}$$

We are now in a position to define the Hopf-Lax formula.

Definition 4.4.2. *Let H be convex and g be Lipschitz. The Hopf-Lax formula is given by*

$$u(\boldsymbol{x}, t) = \min_{\boldsymbol{y} \in \mathbb{R}^d} \left\{ tH^* \left(\frac{\boldsymbol{x} - \boldsymbol{y}}{t} \right) + g(\boldsymbol{y}) \right\}. \tag{4.11}$$

The implementation of this formula in the discrete setting is discussed in Section 4.5.2.

4.5 Discrete Setting

As in Lee et al. [LDO17], we treat the problem as root finding and use the secant method. However, unlike Lee et al. we are specifically interested in redistancing grid based functions. Thus we assume that the initial function is defined in terms of its interpolated values from grid nodes with bilinear interpolation kernel. If we denote the kernel associated with grid node $\boldsymbol{x}_i$ as $N_i(\boldsymbol{x})$, then we may write

$$\phi^0(\boldsymbol{x}) = \sum_i \phi_i^0 N_i(\boldsymbol{x}), \quad \text{and} \quad \phi(\boldsymbol{x}) = \sum_i \phi_i N_i(\boldsymbol{x}), \tag{4.12}$$

with

$$\phi_i^0 = \phi^0(\boldsymbol{x}_i) \quad \text{and} \quad \phi_i = \phi(\boldsymbol{x}_i). \tag{4.13}$$

Also, when we solve for the redistanced values, we do so only at grid nodes (i.e. we solve for $\phi_i = \phi(\boldsymbol{x}_i) = \hat{t}$).

4.5.1 Secant method

To find the roots of $\phi(\boldsymbol{x}_i, \hat{t}) = 0$, we use secant method. Since we only solve for the signed distance value at the grid nodes, the secant method only requires the evaluation of the function $\phi(\boldsymbol{x}_i, t^k)$ for iterative approximation $t^k \to \hat{t}$. We next discuss the practical implementation of the secant method and evaluation of $\phi(\boldsymbol{x}_i, t^k)$ for grid based data.

In order to use the secant method to solve for the root in this context we use the iterative update

$$t^{k+1} = t^k - \phi(\boldsymbol{x}_i, t^k)\frac{t^k - t^{k-1}}{\phi(\boldsymbol{x}_i, t^k) - \phi(\boldsymbol{x}_i, t^{k-1})}. \tag{4.14}$$

The initial guess t^0 can either be set from neighboring nodes that have been recently updated, or generally from a priori estimates of the distance (see Section 4.6.1). However, when no such information is possible or when it would negatively effect parallel performance we use $t^0 = 0$. We set $t^1 = t^0 + \tilde{h}$ where $\tilde{h}$ is proportionate to the grid cell size.

The main concern with using the secant method in this context is that while $\phi(\boldsymbol{x}_i, t)$ is monotonically decreasing in t, it is not strictly monotonic. This means that there can be times t where $\frac{d}{dt}\phi(\boldsymbol{x}_i, t) = 0$. For example, if the minimum of $\phi^0(\boldsymbol{x}_i)$ over the ball centered at $\boldsymbol{x}_i$ of radius t is in the interior of the ball (at a point of distance s from $\boldsymbol{x}_i$), then $\frac{d}{dt}\phi(\boldsymbol{x}_i, r) = 0$ for $s \le r \le t$ (see Section 4.5.2). The secant method is not well defined if we have iterates with equal function values. To compensate for this, if secant would divide by zero, and we have not already converged, we simply increase or decrease $t^{k+1} = t^k \pm \Delta t_{\max}$ in the correct direction. The correct direction is trivial to find, because if $\phi(\boldsymbol{x}_i, t^k) > 0$ then we need to increase t^k. Otherwise we need to decrease t^k. In practice, we use $\Delta t_{\max}$ proportionate to a grid cell size.

Another issue is that errors in the approximation of $\phi(\boldsymbol{x}_i, t^k)$ can lead to more secant iterations. This can be reduced by solving for $\phi(\boldsymbol{x}_i, t^k)$ to a higher tolerance. However, requiring more iterations to approximate $\phi(\boldsymbol{x}_i, t^k)$ more accurately can be more costly than just using more secant iterations with a less accurate (but more efficient) approximation to $\phi(\boldsymbol{x}_i, t^k)$.

For a given tolerance parameter ϵ and ϵ_t, our modified secant algorithm is summarized in Algorithm 1.

Algorithm 1 Modified Secant Method

```
while $|\phi(\boldsymbol{x}_i, t^{k+1})| > \epsilon$ do
    $\Delta t = -\phi(\boldsymbol{x}_i, t^k) \frac{t^k - t^{k-1}}{\phi(\boldsymbol{x}_i, t^k) - \phi(\boldsymbol{x}_i, t^{k-1})}$
    if $|\Delta t| > \epsilon_t$ then
        if $\phi(\boldsymbol{x}_i, t^k) > 0$ then
            $\Delta t = \Delta t_{\max}$
        else
            $\Delta t = -\Delta t_{\max}$
    $t^{k+1} = t^k + \Delta t$
```

4.5.2 Solving Hopf-Lax formula

As mentioned in Section 4.4, for a given time t^k, the Hopf-Lax formula gives us a strategy to compute $\phi(\boldsymbol{x}_i, t^k)$:

$$\phi(\boldsymbol{x}_i, t^k) = \min_{\boldsymbol{y} \in \mathbb{R}^d} \left\{ \phi^0(\boldsymbol{y}) + t^k H^* \left(\frac{\boldsymbol{x}_i - \boldsymbol{y}}{t^k} \right) \right\}, \tag{4.15}$$

or equivalently

$$\phi(\boldsymbol{x}_i, t^k) = \min_{\boldsymbol{y} \in B(\boldsymbol{x}_i, t^k)} \phi^0(\boldsymbol{y}), \tag{4.16}$$

where $B(\boldsymbol{x}_i, t^k)$ is the ball of radius t^k around grid node $\boldsymbol{x}_i$. Thus the problem of evaluating $\phi(\boldsymbol{x}_i, t^k)$ amounts to finding the minimum of the initial ϕ^0 over a ball. While Lee et al [LDO17] use Split Bregman iteration to solve this, we instead simply use projected gradient descent. We used a few hundred projected gradient iterations in practice since this was faster than Split Bregman in parallel implementations due to its relative simplicity. Using $\boldsymbol{y}_k^0$ as an initial guess for the argmin of ϕ^0 over the ball $B(\boldsymbol{x}_i, t^k)$, we iteratively update the approximation from

$$\tilde{\boldsymbol{y}}_k^{j+1} = \boldsymbol{y}_k^j - \gamma \nabla \phi^0(\boldsymbol{y}_k^j) \tag{4.17}$$

$$\boldsymbol{y}_k^{j+1} = \underset{B(\boldsymbol{x}_i, t^k)}{\texttt{PROJ}} (\tilde{\boldsymbol{y}}_k^{j+1}) \tag{4.18}$$

where

$$\underset{B(\boldsymbol{x}_i, t^k)}{\texttt{PROJ}} (\boldsymbol{y}) = \begin{cases} \boldsymbol{y} & \|\boldsymbol{x}_i - \boldsymbol{y}\|_2 \leq t^k, \\ \boldsymbol{x}_i - t^k \frac{\boldsymbol{x}_i - \boldsymbol{y}}{\|\boldsymbol{x}_i - \boldsymbol{y}\|_2} & \text{otherwise.} \end{cases} \tag{4.19}$$

In practice, we set the step size γ equal to the grid spacing Δx. Note that the gradients $\nabla \phi^0(\boldsymbol{y}_k^j)$ are computed using the bilinear interpolation kernels $N_j(\boldsymbol{x})$ as $\nabla \phi^0(\boldsymbol{y}_k^j) = \sum_j \phi_j^0 \nabla N_j(\boldsymbol{y}_k^j)$. We emphasize that for efficiency the sum over j can be taken over only the four grid nodes surrounding the cell that the argument $\boldsymbol{y}_k^j$ is in. We further note that the index for the cell containing the argument $\boldsymbol{y}_k^j$ can be found in constant time using floor$(\frac{y_{\alpha k}^j}{\Delta x})$ where $y_{\alpha k}^j$ are the components of $\boldsymbol{y}_k^j$.

In general, ϕ^0 is a non-convex function defined from the grid interpolation and projected gradient descent will only converge to a local minimum. We illustrate this in Figure 4.1. Failure to converge to the global minimum can lead to large errors in the approximation

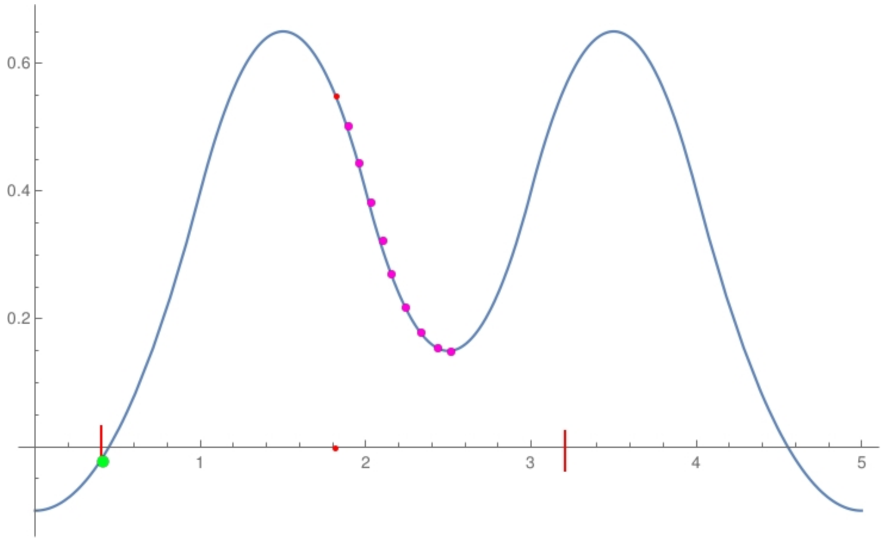

Figure 4.1: ***Local minima illustration.*** *The two vertical lines are the boundary of minimization. Grid node $x_i = 1.8$ is in the middle of the region, and also the starting guess for projected gradient descent. The sequence of points leading off to the right represent the subsequent steps of gradient descent. These points converge to the incorrect argmin $x = 2.5$. The correct solution is at $x = 0.4$. In order to converge to this point, the initial guess would have to be less than* 1.25

of $\phi(\boldsymbol{x}_i, t^k)$. In order to guarantee that we find the global minimizer for the problem, we randomly use multiple initial guesses and take the minimum over the local minima that they determine. In practice, on the order of one guess per grid cell in the ball $B(\boldsymbol{x}_i, t^k)$ is sufficient. For problems without many local extrema the number of initial guesses can be reduced. In general when finding $\phi(\boldsymbol{x}_i, t^k)$ we use as initial guesses $\underset{B(\boldsymbol{x}_i,t^k)}{\texttt{PROJ}}(\boldsymbol{y}_{k-1})$ where

$$\boldsymbol{y}_{k-1} = \underset{\boldsymbol{y}\in B(\boldsymbol{x}_i,t^{k-1})}{\operatorname{argmin}} \phi^0(\boldsymbol{y}), \tag{4.20}$$

is the argmin of ϕ^0 used in the previous secant iteration as well as a small number of random points in $B(\boldsymbol{x}_i, t^k)$. We use this strategy because $\underset{B(\boldsymbol{x}_i,t^k)}{\texttt{PROJ}}(\boldsymbol{y}_{k-1})$ tends to be a good guess for the global minimum. In general, it is very likely that at the next step, the minimum will either be the same point, or along the boundary. Therefore, we prioritize random initial guesses near the boundary of the ball. In fact, for $t^{k-1} < t^k$ we know that the argmin will be at a distance s from $\boldsymbol{x}_i$ with $t^{k-1} \le s \le t^k$ so in theory we should only sample in the annulus.

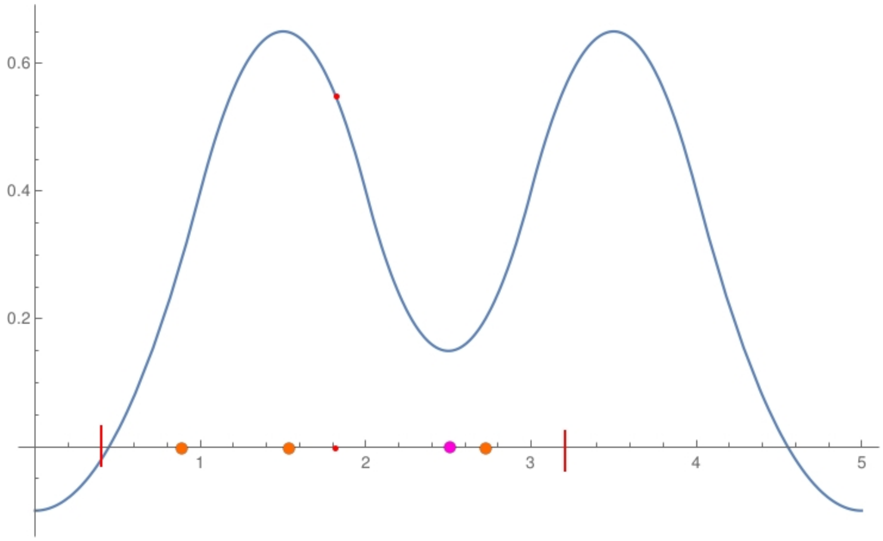

Figure 4.2: ***Random initial guess in 1D.*** *The figure illustrates representative random initial guesses used in solving for $\phi(\boldsymbol{x}_i, t^k)$. In addition we use an initial guess equal to the minimizer computed in the previous secant iteration shown in magenta.*

However, in practice we do not have full confidence in the argmin attained at iteration $k-1$ since our procedure is iterative. Allowing for initial guesses at some locations closer to $\boldsymbol{x}_i$ than t^{k-1} admits the possibility of finding a more accurate argmin. Thus, we found that skewing the sampling density to be higher towards the boundary of the ball struck a good balance between efficiency and accuracy. We illustrate this process in Figure 4.4.

Failure to find the global minimum over the ball can cause unpredictable behavior in the secant iteration for $\hat{t}$. This includes false positives where a t^k is incorrectly interpreted as a root. However, continuing to run secant to a fixed large number of iterations usually corrects for this. In general, there is a tradeoff between the number of initial guesses and iterations of projected gradient and the number of secant iterations. We illustrate this in Figure 4.3 which shows the path to convergence for a few choices of iteration parameters. When $\phi(\boldsymbol{x}_i, t^k)$ is solved with high accuracy, the secant iteration converges with minimal overshoot in 7 iterations. When $\phi(\boldsymbol{x}_i, t^k)$ is not solved to high accuracy, secant overshoots by a large margin, and takes 16 iterations to converge, but notably still converges. However because each iteration is cheaper, the total runtime is lower to reach the same convergence

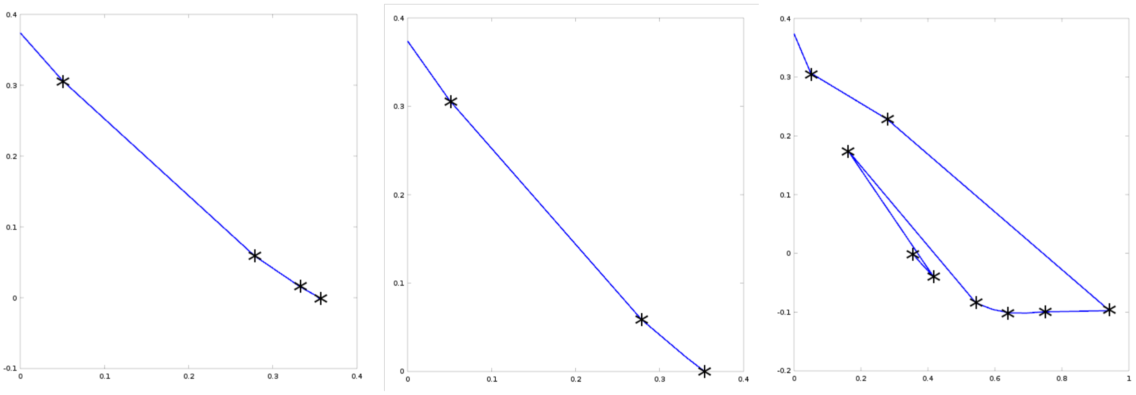

Figure 4.3: ***Different choice of random guesses and gradient discent iteration.*** *The plots above are the points $(\phi(\boldsymbol{x}_i, t^k), t^k)$ found when running our algorithm with different choices of random guess and gradient descent iterations on circle initial data. The left most plot was run with* 100 *random guesses, and* 1 *gradient descent iteration. The middle plot was run with* 1 *random guess, and* 100 *gradient descent iterations. The right plot was run with* 1 *random guess and* 5 *gradient descent iterations. Note that in all cases, the correct root was found.*

for $\hat{t}$. In practice we found that a few hundred projected gradient iterations combined with our initial guess sampling strategy struck a good balance between accuracy and efficiency.

4.6 Results and discussions

4.6.1 Computing in a narrow band

In many applications, only data close to the interface is needed. Since each grid node can be solved for independently, the Hopf-Lax approach naturally lends itself to narrow banding strategies for these applications. We provide a narrow banding strategy based on a coarse

Problem	`num_secant`	`num_rand`	`num_proj`	Timing (ms)
Circle	10	5	100	47.567
Two Points	10	5	100	45.199
Vortex(Per Frame)	10	5	200	73.248
Square	10	4	200	67.582
Sine	10	5	200	71.429

Table 4.1: ***Parameters and timings.*** *Table of parameters for examples and timings.*

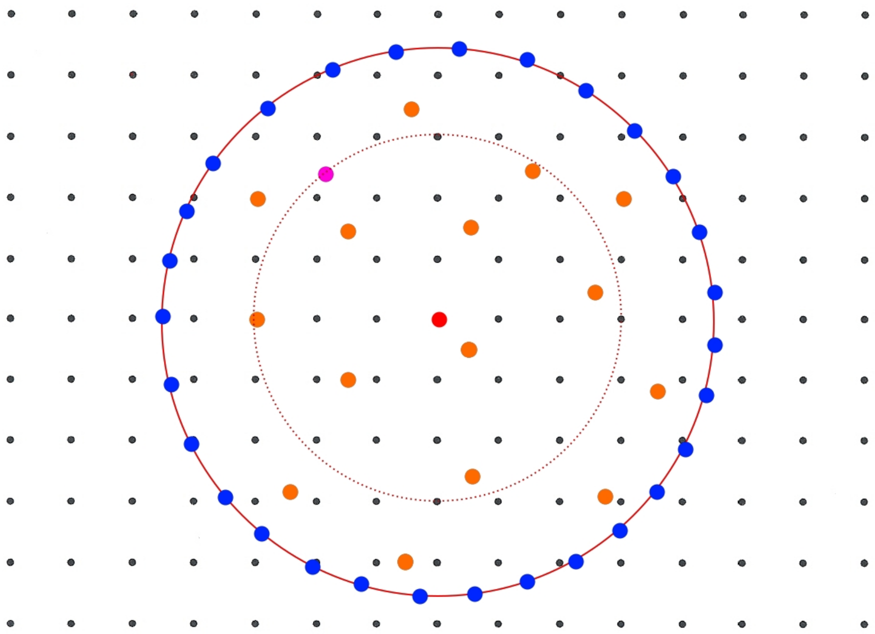

Figure 4.4: ***Random initial guesses in 2D.*** *In this image, the red dot in the center is* $\boldsymbol{x}_i$*, the solid red line represents the ball of radius* t_k *and the dotted line represents the ball of radius* t_{k-1}*. The magenta point was the approximate argmin* $\boldsymbol{y}_{k-1}$ *of* ϕ^0 *over the ball of radius* t_{k-1}*. Since it is unlikely for the minimizer to be inside of* t_{k-1} *we use coarse (random) grid initial guesses in the interior. However, since it is possible that expanding* t *will move the minimizer to a different location we take a large number of initial guesses along the boundary of* t_k

initial grid computation followed by a fine grid update in the narrow band. We first redistance the function on the coarse grid, then we interpolate values from the coarse grid to the fine grid. We then only recompute values on the fine grid that are smaller than a threshold value and we use the value interpolated from the coarse nodes as an initial guess t_0 for the computation on the fine grid. As an example see Figure 4.5.

4.6.2 Computing geometric quantities

The Hopf-Lax formulation naturally allows us to compute normals ($\boldsymbol{n} = \nabla\phi$) and curvatures ($\nabla \cdot \boldsymbol{n}$) at the grid nodes. As pointed out in Lee et al. [LDO17], as the argmin $\boldsymbol{y}^k$ from Equation (4.17) is iterated to convergence, it approaches the closest to point to the grid node $\boldsymbol{x}_i$ on the zero isocontour of ϕ^0. Therefore, recalling that when t^k has converged

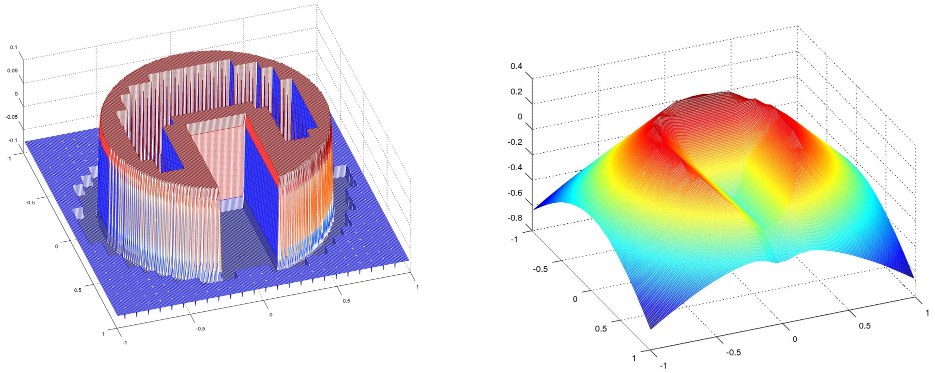

Figure 4.5: ***Narrow band and coarse grid.*** *A coarse grid* 8 *times smaller than the fine grid was solved for initially. Using those values the fine grid was only solved on cells where the distance to the boundary could be less than 0.1 represented as the solid areas of the left image. In the right image those coarse areas are defined from bilinear interpolation. This coarse/banding approach provided approximately a* 2.5 *times increase in performance.*

(within a tolerance) to the root $\hat{t}$ of $\phi(\boldsymbol{x}, \hat{t}) = 0$, then t^k is approximately the distance to the zero isocontour, we can compute the unit normal at the grid node from

$$\boldsymbol{n}(\boldsymbol{x}_i) = \frac{\boldsymbol{x}_i - \boldsymbol{y}^k}{t^k}. \tag{4.21}$$

Notably, this approximation is very close to the exact signed distance function with zero isocontour determined by the bilinearly interpolated ϕ^0. It is as accurate as the argmin $\boldsymbol{y}^k$ so it essentially only depends on the accuracy of the secant method. We get this very accurate geometric information for free. Moreover, the curvature $(\nabla \cdot \boldsymbol{n})$ can be computed accurately by taking a numerical divergence of $\boldsymbol{n}(\boldsymbol{x}_i)$ that would have accuracy equal to the truncation error in the stencil (since no grid-based errors are accumulated in the computation of $\boldsymbol{n}(\boldsymbol{x}_i)$).

4.7 Results

All of the following results were run on an Intel 6700k processor with an Nvidia GTX 1080. The domain for each problem was set to be $[0,1] \times [0,1]$ and was run on a 512×512 grid.

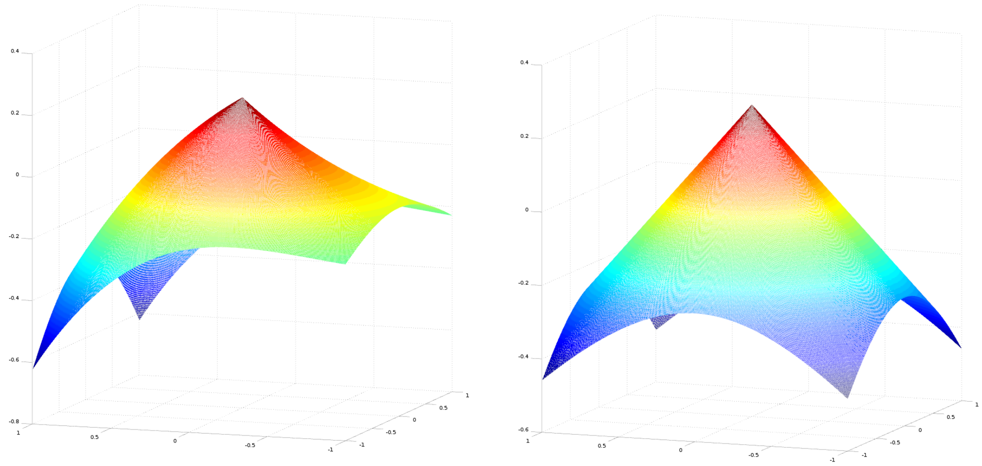

Figure 4.6: ***Scaled circle.*** *The initial data is* $\phi^0 = e^x(0.125 - (0.5 - x)^2 + (0.5 - y)^2)$.

To ensure efficient performance on the GPU, both projected gradient descent and the secant method were run for a fixed number of iterations rather than to a specific tolerance.

Figure 4.6 shows initial data ϕ^0 with a zero-isocontour given by a circle with radius 0.25. Figure 4.7 shows a more complicated test. The zero-isocontour is a square bounded between $[0.25, 0.75]$ in x and y. The corners present rarefaction fans, and the inside needs to handle sharp corners along the diagonals. Because of these difficulties (especially the sharp gradient in our original interpolated ϕ^0), more work is needed in resolving the projected gradient descent step to ensure quick convergence of secant method. The zero-isocontour shown in Figure 4.8 is the union of two overlapping circles. Like in Figure 4.6 the gradient is fairly smooth and thus requires less computation to successfully converge in gradient descent.

In Figure 4.9 we demonstrate our algorithm with a large number of local minima. This problem requires more projected gradient iterations than the simpler examples.

4.7.1 Scaling

All of the following problems were run with the square given in Figure 4.7 with the same parameters. The poor scaling at the low resolutions is due to not using all of the threads possible on the GPU.

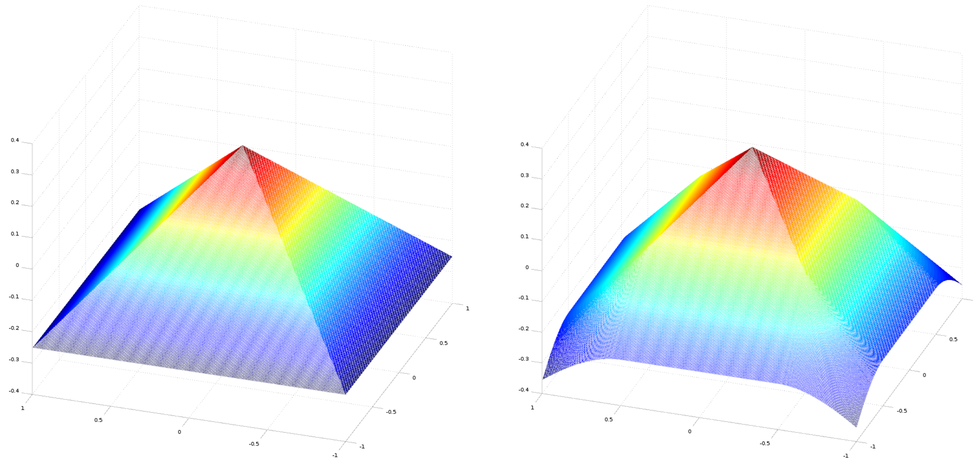

Figure 4.7: ***Square****. The initial data is* $\phi^0 = min\{0.25 - |x - 0.5|, .25 - |y - 0.5|\}$.

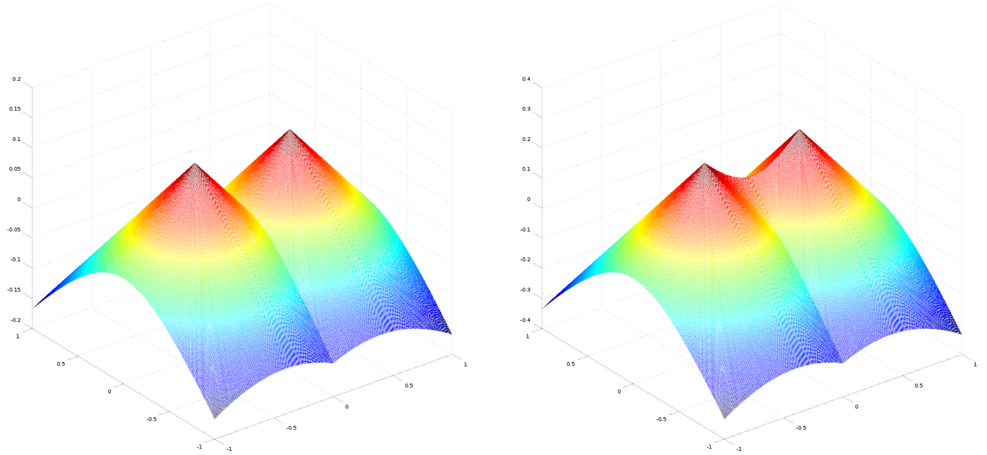

Figure 4.8: ***Union of circles.*** *The initial data is* $\phi^0 = max\{0.25 - \|(0.3, 0.5) - (x, y)\|_2, 0.25 - \|(0.7, 0.5) - (x, y)\|_2\}$.

4.7.2 Level set advection

Figure 4.10 shows our method being used in a level set advection scheme using a simple vortex test. Like previous problems it was run on a 512 × 512 grid. For this problem the average time per frame for redistancing was 73.248 ms.

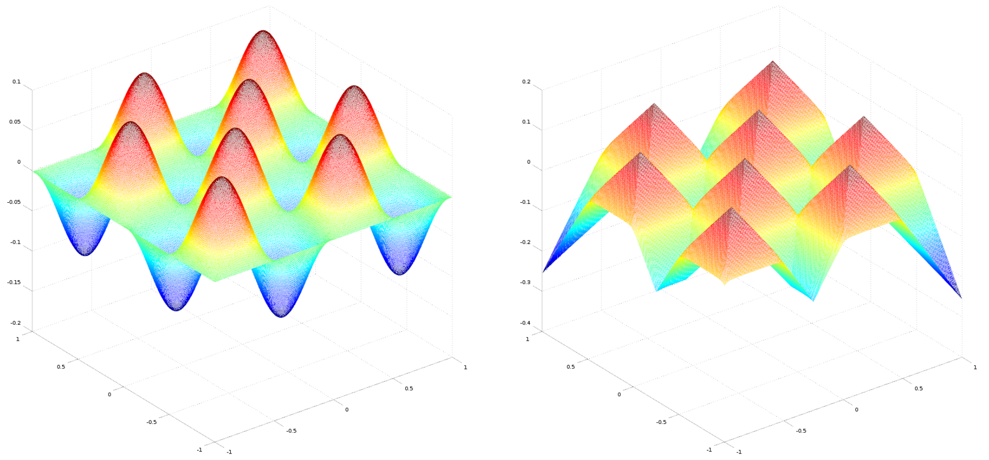

Figure 4.9: ***Many local minima.*** *the objective function is* $\phi^0 = \sin(4\pi x)\sin(4\pi y) - 0.01$.

Size	Timing (ms)	Average ℓ_2 error
32×32	3.303	6.67×10^{-5}
64×64	3.392	1.59×10^{-5}
128×128	4.477	3.87×10^{-6}
256×256	17.840	9.74×10^{-7}
512×512	67.533	2.46×10^{-7}
1024×1024	274.216	6.43×10^{-8}
2048×2048	1185.870	2.38×10^{-8}

Table 4.2: ***Error.*** *Average* ℓ_2 *error with different grid resolution.*

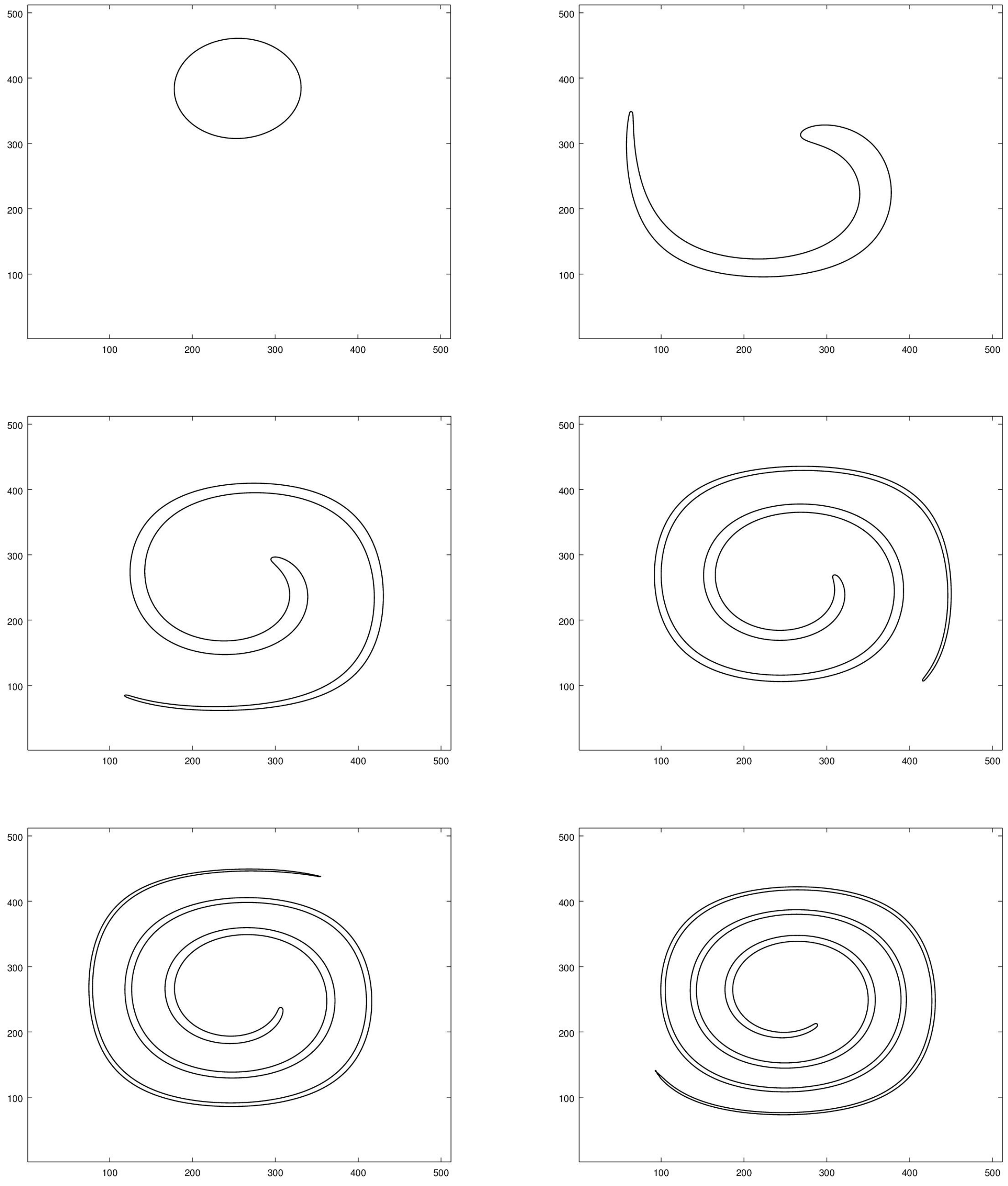

Figure 4.10: ***Vortex test.*** *Vortex advection test at* $t = 0, 1, 2, 3, 4, 5$.

References

[AC76] R. Atkin and R. Craine. "Continuum theories of mixtures: basic theory and historical development." *Quart. J. Mech. App. Math.*, **29**(2):209–244, 1976. 52

[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. 3, 18, 51

[ASB14] K. Abe, K. Soga, and S. Bandara. "Material Point Method for Coupled Hydromechanical Problems." *J. Geotech. Geoenv. Eng.*, **140**(3):04013033, 2014. 51

[ATO09] I. Alduán, A. Tena, and M. Otaduy. "Simulation of high-resolution granular media." In *Proc. Cong. Español Inf. Graf.*, 2009. 4

[BCG11] M. Breuß, E. Cristiani, P. Gwosdek, and O. Vogel. "An adaptive domain-decomposition technique for parallelization of the fast marching method." *App. Math. Comp.*, **218**(1):32–44, 2011. 73

[BFL16] S. Bandara, A. Ferrari, and L. Laloui. "Modelling landslides in unsaturated slopes subjected to rainfall infiltration using material point method." *Int. J. Num. Anal. Meth. Geomech.*, **40**(9):1358–1380, 2016. 51, 61

[Bor06] R. Borja. "On the mechanical energy and effective stress in saturated and unsaturated porous continua." *Int. J. Solids Struct*, **43**:1764–1786, 2006. 52, 60

[BS15] S. Bandara and K. Soga. "Coupling of soil deformation and pore fluid flow using material point method." *Comp. Geotech.*, **63**:199–214, 2015. xiii, 51, 60, 62, 68

[BT07] M. Becker and M. Teschner. "Weakly compressible SPH for free surface flows." In D. Metaxas and J. Popovic, editors, *ACM SIGGRAPH/Eurograph. Symp. Comp. Anim.* The Eurographics Association, 2007. 59

[BW08] J. Bonet and R. Wood. *Nonlinear continuum mechanics for finite element analysis.* Cambridge University Press, 2008. 7, 12, 28, 29, 32

[BWZ10] K. Bao, X. Wu, H. Zhang, and E. Wu. "Volume fraction based miscible and immiscible fluid animation." *Comp. Anim. Virtual Worlds*, **21**(3-4):401–410, 2010. 51

[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. 3, 4

[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. 3

[DB16] G. Daviet and F. Bertails-Descoubes. "A Semi-implicit Material Point Method for the Continuum Simulation of Granular Materials." *ACM Trans. Graph.*, **35**(4):102:1–102:13, 2016. 18, 51

[DE14] F. Dang and N. Emad. "Fast iterative method in solving eikonal equations: a multi-level parallel approach." *Proc. Comp. Sci.*, **29**:1859–1869, 2014. 74

[DG16] M. Detrixhe and F. Gibou. "Hybrid massively parallel fast sweeping method for static Hamilton–Jacobi equations." *J. Comp. Phys.*, **322**:199–223, 2016. 73

[DGM13] M. Detrixhe, F. Gibou, and C. Min. "A parallel fast sweeping method for the eikonal equation." *J. Comp. Phys.*, **237**:46–55, 2013. 73

[Dij59] E. W. Dijkstra. "A note on two problems in connexion with graphs." *Numer. Math.*, **1**(1):269–271, 1959. 72

[DK15] S. Dunatunga and K. Kamrin. "Continuum modelling and simulation of granular flows through their many phases." *J. Fluid Mech.*, **779**:483–513, 2015. 33

[DO16] J. Darbon and S. Osher. "Algorithms for overcoming the curse of dimensionality for certain Hamilton–Jacobi equations arising in control theory and elsewhere." *Res. Math. Sci.*, **3**(1):19, 2016. iii, 74

[DP52] D. Drucker and W. Prager. "Soil mechanics and plasticity analysis or limit design." *Quart. App. Math.*, **10**:157–165, 1952. 2

[Dru00] D. Drumheller. "On theories for reacting immiscible mixtures." *Int. J. Eng. Sci.*, **38**(3):347 – 382, 2000. 61

[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. 44

[Her03] M Herrmann. "A domain decomposition parallelization of the fast marching method." Technical report, DTIC Document, 2003. 73

[HWZ15] X. He, H. Wang, F. Zhang, H. Wang, G. Wang, K. Zhou, and E. Wu. "Simulation of Fluid Mixing with Interface Control." In *Proc. ACM SIGGRAPH/Eurograph. Symp. Comp. Anim.*, pp. 129–135. ACM, 2015. 51

[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. 3, 18, 51

[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. 2, 22, 35, 51, 64

[JSV13] I. Jassim, D. Stolle, and P. Vermeer. "Two-phase dynamic analysis by material point method." *Int. J. Num. Anal. Meth. Geomech.*, **37**:2502–2522, 2013. 51, 62

[JW07] W. Jeong and R. Whitaker. "A fast eikonal equation solver for parallel systems." In *SIAM Conf. Comp. Sci. Eng.* Citeseer, 2007. 73, 74

[KGP16] G. Klár, T. Gast, A. Pradhana, C. Fu, C. Schroeder, C. Jiang, and J. Teran. "Drucker-prager Elastoplasticity for Sand Animation." *ACM Trans. Graph.*, **35**(4):103:1–103:12, 2016. 1, 47, 51

[KPN10] N. Kang, J. Park, J. Noh, and S. Shin. "A Hybrid Approach to Multiple Fluid Simulation using Volume Fractions." *Comp. Graph. Forum*, **29**(2):685–694, 2010. 51

[LD09] T. Lenaerts and P. Dutré. "Mixing Fluids and Granular Materials." *Comp. Graph. Forum*, **28**(2):213–218, 2009. 3, 50, 69

[LDO17] B. Lee, J. Darbon, S. Osher, and M. Kang. "Revisiting the redistancing problem using the Hopf–Lax formula." *J. Comp. Phys.*, **330**:268–281, 2017. iii, 74, 77, 80, 84

[LHM95] A. Luciani, A. Habibi, and E. Manzotti. "A multi-scale physical model of granular materials." In *Proc. Graph. Int.*, pp. 136–146, 1995. 3

[LTK08] F. Losasso, J. Talton, N. Kwatra, and R. Fedkiw. "Two-Way Coupled SPH and Particle Level Set Fluid Simulation." *IEEE Trans. Visu. Comp. Graph.*, **14**(4):797–804, 2008. 51

[LWG08] S. Liu, Z. Wang, Z. Gong, and Q. Peng. "Simulation of Atmospheric Binary Mixtures Based on Two-fluid Model." *Graph. Mod.*, **70**(6):117–124, 2008. 51

[MAM14a] 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. 2, 40

[MAM14b] C. Mast, P. Arduino, G. Miller, and M. Peter. "Avalanche and landslide simulation using the material point method: flow dynamics and force interaction with structures." *Comp. Geosci.*, **18**(5):817–830, 2014. 51

[MAS10] P. Mackenzie-Helnwein, P. Arduino, W. Shin, J. Moore, and G. Miller. "Modeling strategies for multiphase drag interactions using the material point method." *Int. J. Num. Meth. Eng.*, **83**(3):295–322, 2010. xiii, 52, 61, 62, 68

[Mas13] C. M. Mast. *Modeling landslide-induced flow interactions with structures using the material point method.* PhD thesis, University of Washington, 2013. 2, 11, 16, 49

[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. 3

[Mil96] V. Milenkovic. "Position-based physics: simulating the motion of many highly interacting spheres and polyhedra." In *Proc. SIGGRAPH*, pp. 129–136, 1996. 3

[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. 4, 48

[MMS09] V. Mihalef, D. Metaxas, and M. Sussman. "Simulation of two-phase flow with sub-scale droplet and bubble effects." *Comp. Graph. Forum*, **28**(2):229–238, 2009. 51

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

[Mus13] K. Museth. "VDB: High-resolution Sparse Volumes with Dynamic Topology." *ACM Trans. Graph.*, **32**(3):27:1–27:22, July 2013. 68

[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. 3, 18, 48, 51

[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. 3

[NO13] M. Nielsen and O. Osterby. "A Two-continua Approach to Eulerian Simulation of Water Spray." *ACM Trans. Graph.*, **32**(4):67:1–67:10, 2013. 51

[NW06] J. Nocedal and S.J. Wright. *Numerical Optimization.* Springer series in operations research and financial engineering. Springer, 2006. 42

[OS88] S. Osher and J. A. Sethian. "Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations." *J. Comp. Phys.*, **79**(1):12–49, 1988. 71

[Pea86] D. Peachey. "Modeling Waves and Surf." *SIGGRAPH Comp. Graph.*, **20**(4):65–74, 1986. 50

[PGK17] A. Pradhana-Tampubolon, T. Gast, G. Klár, C. Fu, J. Teran, C. Jiang, and K. Museth. "Drucker-Prager Elastoplasticity for Sand Animation." *ACM Trans. Graph.*, **36**(4), July 2017. 1, 42

[RC12] T. Rüberg and F. Cirak. "Subdivision-stabilised immersed b-spline finite elements for moving boundary flows." *Comp. Meth. Appl. Mech. Eng.*, 2012. 58

[RC14] T. Rüberg and F. Cirak. "A fixed-grid b-spline finite element technique for fluid-structure interaction." *Int. J. Num. Meth. Fluids*, 2014. 58

[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. 51

[RJL15] B. Ren, Y. Jiang, C. Li, and M. Lin. “A simple approach for bubble modelling from multiphase fluid simulation.” *Comp. Vis. Media*, **1**(2):171–181, 2015. 51

[RLY14] B. Ren, C. Li, X. Yan, M. Lin, J. Bonet, and S. Hu. “Multiple-Fluid SPH Simulation Using a Mixture Model.” *ACM Trans. Graph.*, **33**(5):171:1–171:11, 2014. 51

[RPL17] M. Royston, A. Pradhana, B. Lee, Y. T. Chow, W. Yin, and S. Osher. *Submitted*, 2017. 1

[RS13] D. Robert and K. Soga. “Soil-Pipeline Interaction in Unsaturated Soils.” In Lyesse Laloui, editor, *Mechanics of Unsaturated Geomaterials*, chapter 13, pp. 303–325. Wiley Online Library, 2013. 61, 62, 67, 70

[RSK08] W. Rungjiratananon, Z. Szego, Y. Kanamori, and T. Nishita. “Real-time Animation of Sand-Water Interaction.” *Comp. Graph. Forum*, **27**(7):1887–1893, 2008. 50, 69

[SCS94] D. Sulsky, Z. Chen, and H. Schreyer. “A particle method for history-dependent materials.” *Comp. Meth. App. Mech. Eng.*, **118**(1):179–196, 1994. 2

[Set96] J. A. Sethian. “A fast marching level set method for monotonically advancing fronts.” *Proc. Nat. Acad. Sci.*, **93**(4):1591–1595, 1996. 72

[SHS12] A. Stomakhin, R. Howes, C. Schroeder, and J. Teran. “Energetically consistent invertible elasticity.” In *Proc. Symp. Comp. Anim.*, pp. 25–32, 2012. 55

[SKB08] M. Steffen, R. M. Kirby, and M. Berzins. “Analysis and reduction of quadrature errors in the material point method (MPM).” *Int. J. Numer. Meth. Eng.*, **76**(6):922–948, 2008. 23

[SSC13] A. Stomakhin, C. Schroeder, L. Chai, J. Teran, and A. Selle. “A Material Point Method for snow simulation.” *ACM Trans. Graph.*, **32**(4):102:1–102:10, 2013. 2, 42, 49, 51, 64

[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. 51, 55, 56

[SSK05] O. Song, H. Shin, and H. Ko. “Stable but Nondissipative Water.” *ACM Trans. Graph.*, **24**(1):81–97, 2005. 51

[TFK03] T. Takahashi, H. Fujii, A. Kunimatsu, K. Hiwada, T. Saito, K. Tanaka, and H. Ueki. “Realistic Animation of Fluid with Splash and Foam.” *Comp. Graph. Forum*, **22**(3):391–400, 2003. 51

[Tsi95] John N Tsitsiklis. “Efficient algorithms for globally optimal trajectories.” *IEEE Trans. Auto Cont.*, **40**(9):1528–1538, 1995. 72, 73

[TSS07] N. Thürey, F. Sadlo, S. Schirm, M. Müller-Fischer, and M. Gross. “Real-time Simulations of Bubbles and Foam Within a Shallow Water Framework.” In *Proc. ACM SIGGRAPH/Eurograph. Symp. Comp. Anim.*, pp. 191–198. Eurographics Association, 2007. 51

[YCR15] T. Yang, J. Chang, B. Ren, M. Lin, J. Zhang, and S. Hu. “Fast Multiple-fluid Simulation Using Helmholtz Free Energy.” *ACM Trans. Graph.*, **34**(6):201:1–201:11, 2015. 51

[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. 4

[YLH14] L. Yang, S. Li, A. Hao, and H. Qin. “Hybrid Particle-grid Modeling for Multiscale Droplet/Spray Simulation.” *Comp. Graph. Forum*, **33**(7):199–208, 2014. 51

[Yos03] N. Yoshioka. “A sandpile experiment and its implications for self-organized criticality and characteristic earthquake.” *Earth, planets and space*, **55**(6):283–289, 2003. 46

[YS17] J. Yang and F. Stern. “A highly scalable massively parallel fast marching method for the Eikonal equation.” *J. Comp. Phys.*, **332**:333–362, 2017. 73

[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. 51

[ZB05] Y. Zhu and R. Bridson. “Animating sand as a fluid.” *ACM Trans. Graph.*, **24**(3):965–972, 2005. 3, 51

[Zha05] H. Zhao. “A fast sweeping method for eikonal equations.” *Math. Comp.*, **74**(250):603–627, 2005. 72

[Zha07] H. Zhao. “Parallel implementations of the fast sweeping method.” *J. Comp. Math.*, pp. 421–429, 2007. 73