Coupled Simulation of Fluids and Granular Materials

by

Michael Tao

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Coupled Simulation of Fluids and Granular Materials

Michael Tao
Master of Science
Graduate Department of Computer Science
University of Toronto
2014

This thesis presents a novel method for the simulation of mixed fluids and granular materials. In contrast to the existing technologies in graphics for simulating granular materials, we choose a continuum based approach in order to capture interesting phenomena with a large number of particles. By extending pre-existing models for simulating both fluids and granular materials as continuua, we generate novel phenomonology in our simulations of both fluids and granular materials, including their interactions with each another.

Contents

1 Introduction **5**

1.1 Outline . . . 6

2 Preliminaries **7**

2.1 Constitutive Equations for Newtonian Fluids . . . 7

2.1.1 Conservation of Mass . . . 8

2.1.2 Conservation of Momentum . . . 8

2.1.2.1 Lagrangian and Eulerian Perspectives . . . 8

2.1.3 Forces . . . 9

2.1.3.1 Pressure . . . 9

2.1.3.2 Free Surface Conditions . . . 10

2.1.3.3 Viscosity . . . 10

2.1.4 Darcy Flow . . . 11

2.1.5 Constitutive Equations for Non-Newtonian Fluids . . . 12

2.1.6 Continuum Behavior of Sand . . . 12

2.1.6.1 Unilateral Incompressibility . . . 13

2.1.6.2 Friction . . . 14

2.2 Summary . . . 15

3 Relevant Work **16**

3.1 Fluid Simulation Techniques . . . 16

3.1.1 Eulerian Grid Fluid Simulation . . . 16

3.1.2 Particle Fluid Representations . . . 17

3.1.3 Semi-Lagrangian Fluids . . . 17

3.2 Granular Simulation Techniques . . . 18

3.2.1 Particle Granular Materials . . . 18

3.2.2 Continuum Granular Materials . . . 19

3.3 Porous Flow . . . 19

4 Overview of Simulation Methods **21**

4.1 Operator Choice . . . 21

4.1.1 Cubical Complexes . . . 22

4.2 Levelset Based Hodge Star . . . 22

4.2.0.1 Length of edges . . . 23

4.2.0.2 General Volumes . . . 23
4.2.0.3 Dealing with Degeneracies . . . 24
4.2.1 Pressure Projection . . . 24
4.2.2 Free Surface Boundaries . . . 25
4.3 Darcy Flow . . . 26
4.3.1 Porosity . . . 26
4.3.2 Porosity Based Hodge Star . . . 26
4.3.3 Capillary Forces . . . 26
4.3.3.1 Maintaining Particle Density . . . 27
4.4 Cohesive Granular Materials . . . 28
4.4.1 Isotropic Strain . . . 28
4.4.2 Deviatoric Strain . . . 28
4.4.2.1 Discretization . . . 28
4.4.3 Using Levelset Hodge Star for UIC . . . 29
4.5 Wet Granular Materials . . . 29
4.5.1 Ordering Operations . . . 30
4.5.2 Advecting quantities . . . 30
4.6 Summary . . . 30

5 Technical Details **31**
5.1 Introduction . . . 31
5.2 Modified MPRGP Quadratic Programming Solver . . . 31
5.2.1 Convergence Criterion . . . 32
5.2.2 Projection Methods . . . 32
5.2.3 Notation . . . 32
5.2.4 Steps . . . 33
5.2.5 Performance . . . 33

6 Results **36**
6.1 Implementation Details . . . 36
6.2 Porous Results . . . 36
6.2.1 2D Rendering Artifacts . . . 37
6.3 Combined Fluid and Granular Results . . . 39

7 Concluding Remarks **41**
7.1 Future Work . . . 41
7.1.1 Stability Issues . . . 41
7.2 Conclusion . . . 41

A Calculus of Differential Forms **43**
A.1 Ordinary Integration . . . 43
A.2 Tangent Bundles on Manifolds . . . 43
A.3 Differential Forms . . . 44
A.4 Differentiating Differential Forms . . . 45
A.5 The Hodge Star . . . 46

A.6 Closed and Exact Forms . . . 46
A.6.1 Hodge-Helmholtz Decomposition . . . 47
A.7 Integrating Differential Forms . . . 47
A.7.1 Cube Complexes . . . 47
A.7.2 Chains . . . 48
A.7.2.1 Laplace-de Rham Operator . . . 48
A.7.3 Integration . . . 49
A.8 Discrete Calculus on Differential Forms . . . 49
A.8.1 Exterior Derivative and Boundary Operators . . . 50
A.8.2 Hodge Star . . . 50
A.9 Pressure Projection . . . 50

B Discrete Differential Forms on Cubical Complexes 51
B.1 Level-set based Hodge Star . . . 51
B.1.1 Connection to Staggered Grid Methods . . . 51
B.1.2 Boundary Operator . . . 52
B.1.3 Hodge Star . . . 52
B.1.3.1 Hodge Star on Uniform Grids . . . 52
B.1.3.2 Differential and Codifferential Operator . . . 53
B.1.3.3 Constructing the Laplace-deRham . . . 53

C Numerical Optimization 54
C.0.4 Constrained Optimization . . . 54
C.0.5 Linear Complementary Problems . . . 55
C.0.6 LCP and Karush-Kuhn-Tucker Conditions . . . 56
C.0.7 Complimentary and Minimization Duality . . . 56

Chapter 1

Introduction

Uncountably many films have scenes that contain beaches, but there is rarely, if ever, a proper simulation performed on the large body of sand. That is because simulating the interactions of sand particles is an intractable problem: there are simply too many particles that can interact with one another. Even if we could make the simulation of sand tractable, we would have only shown how to simulate a desert. Without being able to simulate water, and its interactions with sand, we would not have a method for generating beaches.

Sand is a granular material; this family also includes materials like rice and cornstarch, which are ubiquitous materials in human the world. By taking into account the prevalence of water and other fluids on our planet, the combined dynamics of granular materials and fluids are some of the most common phenomena on the planet. In fact there are many recreational human applications of this multi-material interaction such as sand castles and cooking, as well as industrial applications in the design of crucial infrastructures such as water filtration systems and bridges.

Although the combination of granular materials and fluids has such great presence in our lives, there are only a few methods available for simulating them at a scale of more than thousands of particles, which wouldn't be able to even fill a pail of sand in a sandbox. The reason for this lack is multi-fold, but mostly it derives from a variety of representational issues.

The natural way one might think of granular materials is as a collection of its constituent granules. That is, a granular material is a collection of small, relatively rigid, particles that interact with each other through rigid body collisions and conservation of momentum. Though this solution is direct and obvious, it is not easily scalable because the forces generated by any single particle can perturb the whole system, which implies that we would need a global solution for the dynamics of each simulation timestep. Indeed, advecting a body of granules is a global problem, where the force from one particle can easily affect fairly distant particles as one can readily see in an apparatus such as Newton's cradle.

Furthermore, the granule-based approach is not amenable to some standard representations of fluids. If one were to choose a particle-based method the fluid particles would have to be significantly smaller than the size of the granules in order for the fluid particles to be able to advect through a mass of granular particles. With the choice of grid or mesh based methods, there are several issues that arise due to the fundamental differences between the sort of quantities stored. Communication between the granules and the fluid stored on the grid would lead to a fair amount of averaging and interpolation between the grid and the particles.

Now if we instead consider the option of treating granules as a continuum, we are quickly confronted withtwo fundamental issues. The first is that much of the interesting behavior from granular materials comes from the behavior of individual particles, which can quickly disappear as "interesting" simulation scales would require particles to be at such fine scales so as to would make them invisible. The second, far more glaring issue is that there is no general consensus of a single model for granular materials as a continuum. There are too many variables available such as granule distribution and granule geometry to take into account that can change the behavior of the granular material as a whole and the addition of fluids doesn't help with the situation.

In this thesis we discuss how one can ameliorate some of the issues that one encounters through a hybrid simulation method that advects individual granules, and allows for granule-granule interactions, but also takes into account global scale effects through a grid to maintain global effects, which include the addition of fluid matter into the bodies of granules.

1.1 Outline

This thesis is organized as follows:

- Chapter 2 reviews some rheological preliminaries, with Appendix A to provide some further mathematical preliminaries.
- Chapter 3 provides comparison with previous work in the field.
- Chapter 4 overviews of the method in fairly general terms.
- Chapter 5 describes the method in its technical details.
- Chapter 6 finishes the thesis with some results and final remarks.

Chapter 2

Preliminaries

In order to develop an intuition for the nature of granular materials, fluids, their mixtures, and how to simulate them we will first explore some necessary physical and mathematical foundations. This section will provide an overview of the intuition behind the various data structures and equations that are used later. We choose to use the language of exterior calculus and differential forms in our mathematical descriptions. This choice allows us to use a simple and elegant family of linear differential operators that allow for a significant amount of deformation in the space. For completeness we have included a brief overview of the those topics in Appendix A.

The two categories of material we will discuss are what are frequently called Newtonian and a dilatant Non-Newtonian fluid. Though we will go into their definitions in further detail in the following sections, Newtonian fluids are what one normally thinks of as a fluid such as water and oil. Other types of fluid are called Non-Newtonian fluids, where dilatant (or shear thickening) fluids are a specific type of Non-Newtonian fluid that act more like a solid when a moderate amount of pressure is applied to it. In our case we are looking at sand, but dilatancy is a common feature in different types of mixtures such as blood, ketchup, and oobleck

We will first tread through the more commonly known constitutive equations used for Newtonian fluid simulation, then expand on those equations to generate constitutive equations for more general flows.

2.1 Constitutive Equations for Newtonian Fluids

Deriving a model for the dynamics of fluids begins by trying to tracking infinitesimal parcels of fluid material while guaranteeing that basic material laws are followed. In particular, all materials must satisfy the conservation laws of mass and balance of momentum. With just those two conservation laws, viscosity, and incompressibility we can obtain the standard equations used for modeling Newtonian fluids – the Navier-Stokes Equations.

2.1.1 Conservation of Mass

The amount of mass of fluid in a particular volume Ω can be identified by the integral of density ρ defined over Ω through the integral

$$\text{mass}(\Omega) = \int_\Omega \rho dV$$

Conservation of mass is precisely stated by looking at the time derivative of mass:

$$0 = \frac{\partial}{\partial t}\int_\Omega \rho dV = -\int_{\partial\Omega} \rho u \cdot n \; dA.$$

where u is the velocity of mass, n is a the unit outwards normal vector, dV is the volume. Because $u \cdot n$ represents the "amount of velocity going outside on the boundary of Ω," the right hand side can be interpreted as the "amount of density moving outside on the boundary of Ω."

By using the divergence theorem and moving everything to the left hand side we obtain

$$\int_\Omega \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) dV = 0$$

Because this equation applies to any domain $X \subset \Omega$, we can write this integral equtation as a differential equtation:

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

which is called the continuity equation. Quite simply, the change in density at a point is equal to the amount advected through that point, so no material will spontaneously appear.

2.1.2 Conservation of Momentum

Fluids follow the Newton's famous equation for balancing momentum: $F = ma$. When we write it by tracing the forces applied to an infinitessimal parcel of material, treated as a particle:

$$F = \rho \frac{Du}{Dt}$$

where $\frac{D}{Dt}$ is the material derivative, which represents the change in a function for a specific particle. This special derivative takes into account the fact that the derivative we evaluate at a particular point in space/time does not correspond to the derivative of a value on a particular location. For example, if one were to place a depth sensor on a boat and allow it to sail into an ocean, the sensor would probably detect changes in depth, resulting in a nonzero derivative for the depths recorded by the sensor. The material derivative would take into account the velocity of the boat, causing the derivative to be zero.

This virtual particle is best explained through an exposition of the Eulerian and Lagrangian perspectives, as we now discuss.

2.1.2.1 Lagrangian and Eulerian Perspectives

As a fluid is advected in a domain there are two natural ways to record quantities of those fluids as they move: the Lagrangian perspective and the Eulerian perspective. In the Lagrangian viewpoint the fluid is represented as a collection of constituent particles. Dynamics are performed by moving the positions of particles in space through time, similar to the boat in the previous example. The Eulerian viewpoint

maintains measurements of quantities at fixed positions in time, similar to the ground in the previous example.

The material derivative is simply derived by utilizing the chain rule. Taking the derivative of a scalar quantity q being advected along a velocity field u we define the material derivative to be the time derivative of the quantity at a particular point and time:

$$\frac{Dq}{Dt} = \frac{d}{dt}q(t,x) = \frac{\partial q}{\partial t} + \sum_i \frac{\partial q}{\partial x_i}\frac{dx_i}{dt} = \frac{\partial q}{\partial t} + \nabla q \cdot u.$$

The material derivative is split into two terms: the change of a quantity at a point, $\frac{\partial q}{\partial t}$, and the offset caused by the quantity being advected by u, $\nabla q \cdot u$. This second term can be seen as correction required to translate between the Eulerian and Lagrangian perspectives.

Assuming incompressibility is the same as assuming $\frac{D\rho}{Dt} = 0$, we obtain, through subtraction with the continuity equation and the chain rule, that

$$\begin{aligned} 0 &= \frac{D\rho}{Dt} - \frac{\partial \rho}{\partial t} - \nabla \ \cdot (\rho u) \\ 0 &= \frac{\partial \rho}{\partial t} + \nabla\rho \cdot u - \frac{\partial \rho}{\partial t} - \nabla \cdot (\rho u) \\ 0 &= \rho \nabla \cdot u \\ 0 &= \nabla \cdot u. \end{aligned}$$

Such a velocity field is said to be divergence-free. Requiring the velocity field to be divergence-free is often called the incompressibility condition.

2.1.3 Forces

In the Navier-Stokes equations there are three types of forces that direct the motion of fluid: pressure, viscosity, and external forces. The force generated by pressure is the force pushing a fluid from high pressure regions to low pressure regions and the force from viscosity is the fluid equivalent of a frictional force from within the fluid. External forces will generally mean the sum of body forces including gravity and boundary forces such as interactions with rigid bodies.

Computationally, viscosity and pressure are typically separated. This comes from Chorin's splitting method[Cho67], where the computation required to maintain incompressibility and to generate viscous forces are separated. Splitting is crucial to maintain incompressibility, and also allows for the trivial removal of viscosity in situations when simulating fluids that have a low viscosity. These fluids are called inviscid fluids, and are very common in graphics because of water's low viscosity and the cost of numerically solving for viscosity would suggest taht additional computation be used only when necessary.

2.1.3.1 Pressure

In an incompressible fluid, pressure can be seen as the force that maintains the incompressibility constraint, so pressure can be computed as the force returning $\nabla \cdot u$ to zero. After applying the various other forces present in a system, the velocity field for a fluid tends not to be incompressible anymore. A simple way to maintain incompressibility is to find new velocity fields to substitute the invalid velocity

fields, which can be done through a projection in the L^2 sense from the invalid velocity field back into the space of gradient-free vector fields.

We find that the best derivation of this is done with differential forms. In the language of differential forms the incompressibility condition can be written as $du^\flat = 0$. When it is not ambiguous, we will omit the $\cdot^\flat$ from u because the contexts for when u is treated as a 1-form and vector field are unambiguous. From the perspective of differential forms the incompressibility condition is the same as u being co-closed. This is easily seen by considering the Hodge-Helmholtz decomposition A.6.1 that any any differential k-form can be written in terms of a $k-1$-form, $k+1$-form and a k-harmonic form. If we set the 0-form in our case to be labeled as p for pressure, we see that

$$\begin{aligned} \delta u &= \delta dp + \delta\delta\omega + \gamma \\ \delta u &= \delta dp \end{aligned}$$

Therefore $u - dp = \delta\omega + \gamma$ represents a divergence-free vector field, which is therefore incompressible. This definition of pressure treats can be treated as a projection operator of a velocity field u to the L^2-nearest incompressible. The force necessary to set $\nabla \cdot u$ is therefore $-\nabla p$.

For a more traditional vector-calculus perspective, one can check our discussion of differential forms A.4 to see that $\delta u^\flat = \nabla \cdot u$ as well as that $(dp)^\sharp = \nabla p$ to derive

$$\begin{aligned} \delta u &= \delta dp \\ \nabla \cdot u &= \nabla \cdot \nabla p, \end{aligned}$$

which is the standard pressure projection problem.

2.1.3.2 Free Surface Conditions

One crucial component for interesting fluid simulations is to have a visible fluid boundary. In order to produce such a simulation there are some minor changes required in the pressure solving step to take into account the incompressibility of the system while handling the boundary. One standard solution is to set the pressure of the air to be 0, which we will utilize and go into more detail in the Overview 4.

2.1.3.3 Viscosity

Viscosity can be seen as a sort of friction between infinitesimal particles. The force effectively reduces the amount of shearing that the fluid can do, which homogenizes fluid flow. That is, it pushes the velocity field of a fluid to be closer the local average velocity. It is essentially a diffusion of momentum in a fluid. The force from viscosity does not satisfy an intrinsic constraint like incompressibility, but is a force returning to a body to counteract shearing in the material.

Viscosity is most readily defined on the boundary as a tensor σ such that for a point on the boundary, with normal n, the viscosity force is σn. Unlike pressure, where we used a scalar value p, σ has to be a tensor to take into account the fact that the force of viscosity will not be parallel to n.

The local changes in u are represented by ∇u, so viscosity forces must be representable as a function of ∇u. As a first order approximation we will assume that σ is linear with respect to ∇u. Because of constraints such as the symmetry and rigid-body invariance of σ, not all of ∇u is relevant, and it is

possible to remove parts of the operator that have no bearing to the computation of viscosity. First, consider the decomposition

$$\nabla u = \frac{\nabla u + \nabla u^t}{2} + \frac{\nabla u - \nabla u^t}{2} = D + S.$$

Because rigid rotations should not have an affect on the relative velocities, we can ignore S and only pay attention to D.

Because D is symmetric, that is, it is invariant on the order of its vector arguments, it can be diagonalized. In order to be symmetric on each of the axes, the operator from D to σ can only be a scaling of D and an added diagonal term. The only possible linear term to add to the diagonal is a scaling of $\nabla \cdot u$ in order to add the different $d_i u_i$ terms symmetrically. We therefore have a form like

$$\sigma = \lambda \nabla \cdot u \mathbb{I} + 2\mu D$$

Because we are looking at an incompressible fluid, $\nabla \cdot u = 0$, the leftover terms of viscosity will be

$$\sigma' = \mu D.$$

By applying the divergence theorem in the same way utilized for pressure, we see obtain a body term of

$$\mu \nabla \cdot D = \mu \frac{\nabla \cdot \nabla u + \nabla \cdot (\nabla u^t)}{2} = \mu \nabla \cdot \nabla u.$$

The viscous Navier-Stokes equations therefore have the following form:

$$\begin{aligned} \rho \frac{Du}{dt} &= \nabla \cdot \tau - \nabla p + g \\ \nabla \cdot u &= 0 \\ \tau &= \mu(\nabla u + (\nabla u)^T) \end{aligned}$$

where g are all external forces, usually including gravity, and μ is the viscosity coefficient, which is a property of the fluid being simulated.

2.1.4 Darcy Flow

Fluids in a porous medium are by definition surrounded by solid matter so the velocity of a wet porous mass should be approximately the same as the velocity of the particles and porous material. In this sort of flow, which is commonly called Darcy flow, we can therefore assume that the momentum is eliminated by the solid medium. When the porous medium is not moving in space, one can represent the removal of momentum as the Eulerian and Lagrangian perspectives being identical. That is, we will assume that

$$\frac{Dq}{dt} = \frac{\partial q}{dt}$$

in the context of Darcy flow.

We also add a force of capillary action, which is defined by a surface potential that defines the affinity of the surface of granules with whatever fluids are interacting with them. Capillary action is what allows for water to diffuse against the flow of gravity, through porous materials like sand, sponges, and paper.

Though capillary action is a surface-to-surface interaction, at a continuum level it is treatable as a volumetric quantity through homogenization, utilizing some assumptions on the porous body's internal geometry [Bea72]. For instance one can assume that the interior of a sponge is a series of spheres or cylinders to generate different models for the volume to surface area ratio.

With those two changes, we obtain the following equations for incompressible flow:

$$\begin{aligned} \frac{\partial u}{\partial t} &= -\frac{\kappa}{\mu}\nabla p + \frac{\kappa}{\mu}\rho\hat{g} \\ \nabla \cdot u &= 0 \\ \hat{g} &= -\nabla\Phi + g \end{aligned}$$

where κ is permeability, μ is viscosity, and $-\nabla\Phi$ is capillary potential, and g is usually gravity. This same procedure has been seen applied to heat transfer in [**?**]. We will suppress the above fraction with the variable $\hat{\kappa} = \frac{\kappa}{\mu}$.

2.1.5 Constitutive Equations for Non-Newtonian Fluids

The rhelogical behavior of sand is quite different from water. At the continuum level this difference is often mentioned as the differences between a Newtonian and a non-Newtonian fluid, with water being the former and sand being the latter. What determines whether a fluid is Newtonian or not is its response to shearing.

In the case of a Newtonian fluid stress is a linear response to shearing. The behaviors described by this model are the standard ones one imagines for a fluid like water: the resistance one meets when pushing against the fluid is proportional to the relative speed of the fluid itself.

Any fluid with a nonlinear stress/strain relationship is considered non-Newtonian. In the non-Newtonian case, the viscous stresses can depend on a variety of other factors such as pressure, temperature, or electromagnetic effects. The behaviors are much more varied than those of Newtonian fluids. With shear thickening fluids like cornstarch suspensions, beyond some amount of pressure on the fluid the force required to move through the material becomes substantially more difficult. It is even possible to run on a pool filled with cornstarch suspended in water.

On the other hand, shear thinning fluids like ketchup seem somewhat rigid at rest but will become fluildized when pressure is applied. This particular behavior is why many modern condiment containers have caps on the bottom: gravity applies insufficient pressure for the ketchup to fluidize so it does not escape the container, but once some pressure is applied the ketchup will fluidize and squirt out.

2.1.6 Continuum Behavior of Sand

The rheological behavior of granular materials at a continuum scale is that of a time independent viscous non-Newtonian fluid; in particular they are shear thickening. The structure of the relatively rigid individual particles aggregates frictional forces so that, past a specific amount of pressure, a given amount of shear stress will not generate further shearing. Because of its time independence, the flow of a granular material can be locally defined as an energy minimization based on its stress tensor σ which we separate into the isotropic mean stress term (pressure) p and the traceless deviatoric stress term (frictional) s:

$$\sigma = -pI + s.$$

This is similar to the decomposition that we saw before for incompressible fluids, but here we do not force the frictional component to be linear. Sand can be treated as being incompressible, so pressure should be the same linear equations as before, while the s is not due to the existence of two different modes of friction that exist: static and kinetic.

The boundary between the two frictional regimes can be expressed through a yield criterion. The criterion demarks a boundary constraint on the stress tensor before the granular material acts in a rigid regime. Intuitively what we want to represent is that a sufficient amount of shearing stress must occur before any motion happens, and that the yielding point increases monotonically with pressure. Usually the yielding point is linear to pressure. Here we use the Drucker-Prager yield condition, a continuous variant of the Mohr-Coulomb yield condition [ZB05]:

$$\|s\| \leqslant \sqrt{3}\alpha p + c$$

for some norm $\|\cdot\|$ (we will use the Frobenius or L^2 norm) and α, c being material parameters. The term c corresponds to cohesion, which I will assume is 0 to simulate dry granular materials. α has a geometric interpretation: if the steepest angle that granules can be piled, the angle of repose, is θ, then $\alpha = \sqrt{2/3}\sin\theta$.

The method we use alternates between solving for a velocity that satisfies pressure constraints and solving for frictional constraints using a variational principle.

2.1.6.1 Unilateral Incompressibility

In the above framework the static representation of granular material is stored precisely as that of a viscous fluid. The only difference represented is the yield condition that appears in the resolution of the stress. However, by treating granular materials like a viscous fluid, some crucial information on the incompressibility of individual grains is lost. Traditional fluid incompressibility does not assume any atomic set of spherical particles, and each parcel of fluid is allowed to deform into arbitrary shapes and aspect ratios. Granular materials, however, consist of real physical particles that cannot deform and so a standard pressure solve is not sufficient. Resolving particle collisions is a global problem because forces are propagated at a much higher speed than our intended timesteps, so some form of global incompressibility solution is required. Narain et al. [NGCL09] introduce the Unilateral Incompressibility Constraint (UIC) in order to roughly solve this problem at a global scale. The intent is to use continuum methods to find a rough but adequate global solution using a cheap local solver to find a reasonable solution.

Enforcing the Unilateral Incompressibility Condition, which represents the maximal density obtained when packing spheres, at global scale seems to be a decent candidate for the approximate solution we seek. The condition explicitly sets a maximal density

$$\rho_{\max} = 2\alpha/(\sqrt{3}d_{\min}^2)$$

for a parameter α representing imperfections in packing and $d_{\min}$ representing the minimal distance allowed between objects (the radius of a particle of sand). When the density of a grid cell is below the packing density, the system assumes that there is no need to interrupt the advection of particles in that particular grid cell. However, if the particle density increases to beyond the maximal density, the system generates a pressure force to guarantee that by the next timestep the density is lowered to a valid level.

Within a homogeneous medium and given a density constraint, the mass of material in a finite volume can be identified with a volume fraction ϕ, and the momentum of that mass can be identified with ϕv. In [NGCL09] the advection of density through pressure is given by

$$\phi^{n+1} = \phi^{n+1}|_{p=0} - \frac{\Delta t^2}{\rho_{\max}} \nabla^2 p$$

where $\phi^{n+1}|_{p=0}$ is the density at the next time step ignoring pressure. The complimentary constraint $p(1-\phi^{n+1}) = 0$, implies that a nonzero pressure will exist if and only if the system is completely packed in that cell. We can therefore formulate this as a linear complementary problem (LCP):

$$A_1 p + b_1 \geqslant 0 \tag{2.1}$$

$$p \geqslant 0 \tag{2.2}$$

$$p^T (A_1 p + b_1) = 0 \tag{2.3}$$

where

$$A_1 = \frac{\Delta t^2}{\rho_{\max}} {D_1}^T D_1 \tag{2.4}$$

$$b_1 = 1 - \phi^{n+1}|_{p=0} \tag{2.5}$$

given that D_1 is the finite difference gradient operator on the density. For more information on LCP problems refer to Appendix C.

This method is sufficient for guaranteeing that particles do not usually get too close but it is not an exact solution to completely avoiding self intersection. Therefore, after each UIC solve the simulation usually pushes each particle around in a small random walk, which works because the particles are already mostly separated out and only need to be jittered slightly.

2.1.6.2 Friction

Frictional stress is computed by minimizing the kinetic energy in a system:

$$E = \frac{1}{2} \int \rho^n \|v\|^2 dV.$$

For computational purposes a reweighting is performed for $w = \rho^n / \rho_{max}$

$$E = \frac{1}{2} \int w \rho^n \|v\|^2 dV.$$

This modification introduces some error but improves the conditioning of the problem by reducing the variation of coefficients used by low-density portions of the discretized system. The constraints of the Druker-Prager yield criterion are approximated with a series of planar constraints by bounding the coefficients of the tensor by $s_{\max} = \alpha p$ [NGCL09]. Minimizing this energy functional is a quadratic so we obtain the following quadratic system:

$$s^T A_2 s + b_2^T s \geqslant 0 \tag{2.6}$$

$$s_{\max} \geqslant s \geqslant -s_{\max} \tag{2.7}$$

where

$$A_2 = \frac{\Delta t^2}{\rho_{\max}} D_2^T D_2 \tag{2.8}$$

$$b_2 = \frac{\Delta t}{\rho_{\max}} {D_2}^T \rho^n v|_{s=0} \tag{2.9}$$

given that s is a vector of the stress tensor coefficients and D_2 is the finite difference gradient operator on the stress tensor. This system is easily transformed into a linear complementary problem by applying the Karush-Kuhn-Tucker conditions, which are an extension of λ-multipliers that takes into account boundary conditions.

2.2 Summary

In this chapter we first derived the necessary equations for describing the dynamics we will discuss in later sections. We first derived standard fluid flow equations to provide some basic insight on fluid simulation and how the dynamics are conventionally separated. From there, we provided alternative models for pressure and for viscosity to present a model for simulating granular materials, including some numerical details. For details on the mathematics we will use or for further information on LCP problems please refer to the appendices (Appendix A and Appendix C respectively).

Chapter 3

Relevant Work

The rheology of non-Newtonian fluids has only recently appeared in computer graphics literature [GBO04], with wet granular materials only appearing as recently as 2008 [RSKN08]. Non-Newtonian fluids in the general sciences are a well known topic, though the precise mechanisms behind how they operate are still open for discussion. There have been some results in non-physical simulation of sand and mud such as the work of [SOH99], but we will primarily pursue the physical aspects of simulation. Cohesive granular materials and porous flow have a strong history in engineering, in areas such as soil mechanics [Har25] but we will focus our scope on simulations of visual phenomenology and thus focus on the computer graphics literature. We primarily restrict our discussion to publications from the domain of graphics.

3.1 Fluid Simulation Techniques

The first attempts to fully simulate the Navier-Stokes equations in graphics literature was by Foster and Metaxas [**?**], but there is a deeper history in the engineering literature. Foster and Metaxas's paper was based on the earlier work of Harlow and Welsh [HW$^+$65] which is the origin of staggered grid methods. In fact, many of the modern fluid simulation techniques have deep histories in engineering.

3.1.1 Eulerian Grid Fluid Simulation

Grid based simulation methods have a long history in computer graphics, starting with the seminal work of Foster and Metaxas mentioned above. That method, and many of its successors, had difficulties with numerical stability with large timesteps where the total energy of a system would explode. The issue lay with the use of the use of an explicit time integrator. It was with Stable Fluids by Stam [Sta99] that a semi-implicit integration method appeared in the graphics literature, which prevented such energy blowups. That guarantee, however, came at the cost of significant numerical viscosity, which caused energy to disappear from a system. Since then multiple methods have become popular in graphics to compensate for numerical viscosity like vorticity confinement [FSJ01].

The methods mentioned above are excellent for defining fluid flow when a static domain is filled with fluid and a few quantities are advected around by a velocity field, as is the case when simulating smoke. However, for many graphics applications, fluid does not fill a static domain, and instead covers a dynamic subdomain of a larger domain. For example, in a scene where water is being poured into a glass, there is initially no water in the interior of the glass and over time the subset of the domain

comprised of water changes dynamically. In order to handle such a scene using a Eulerian method, some extra work is required.

Simulating fluids with boundaries that can form waves and splash are commonly called free surface fluids. Surface tracking technologies like the level set method [EMF02, ZOF01] have become popular because they allow simulations to represent the interface between a fluid and air. Eulerian fluid simulation techniques are exemplary at simulating incompressible free surface flows when properly coupled with levelset data [BBB07]. This is due to the availability of simple differential operators for computing the pressure required to maintain incompressibility. Purely Eulerian methods, however, are inherently limited and have a difficult time with numerical viscosity and handling free surfaces.

3.1.2 Particle Fluid Representations

Particle-based fluid techniques are naturally amenable for simulations involving free surfaces, though the smoothness of the generated surfaces can be difficult to maintain. There has been, however, substantial efforts in that direction like [YT13]. The most popular particle-based method used for fluid simulation, and more recently granular simulation, is Smoothed Particle Hydrodynamics (SPH). This is a method whose origins are in the astrophysics community in work by Gingold and Monaghan [GM77] for tasks in compressible flow. SPH was brought into the graphics literature by the work of Desbrun and Gascuel [DG96] for purely Lagrangian fluid simulation. Beyond representing particles being advected through space, it represents a family of radial basis functions ϕ_i^t around a set of points advected through space. Under this framework, a function f discretized at a time t would be computed by attaching coefficients $\bar{f}_i^t = \langle f, \bar{\phi}_i^t \rangle$ (where $\bar{\phi}_i^t$ is a dual basis function for ϕ_i^t), therefore producing the approximation

$$\bar{f}^t = \sum \langle f, \bar{\phi}_i^t \rangle \phi_i^t = \sum \bar{f}_i^t \phi_i^t.$$

The basis functions do not inherently represent the boundary of a surface and the repulsive forces they do generate are smooth, which naively leads to timestep restrictions and "soft" particles - in that particles act as if they were undergoing elastic deformation. This "soft" particle property is problematic for SPH in that it has trouble maintaining incompressibility in the long run, though work on maintaining a semblance of incompressibility has been an area of active research [BT07, SP09, ICS+13]. Compressibility is a difficult problem to solve in a purely Lagrangian context, but not too difficult to handle in Eulerian methods, so one could imagine trying to alleviate ompression issues by combining both methods. Methods that utilize the advantages of both Eulerian and Lagrangian methods are genreally what are called Semi-Lagrangian methods.

3.1.3 Semi-Lagrangian Fluids

Among the the most succcessful Semi-Lagrangian methods are Stable Fluids and FLIP. FLIP was introduced into the graphics community by Zhu et al. [ZB05]. This method utilizes particles to advect velocities and a grid to solve for pressure and add external forces. Although it suffers from some long term volume loss and numerical viscosity from the large amount of interpolation that is performed , the loss is much less than plain SPH, and the numerical viscosity is also much less than that of a standard purely Eulerian technique. The work of Batty et al. [BBB07] extended the use of FLIP particles to generate a levelset to generate a more accurate Laplacian operator. More recently, the Hybrid SPH

Method [RWT11] by Raveendran et al. have utilized that sort of pressure solve to guide the advection of SPH particles to decrease compression.

Pure levelset surface tracking has evolved to utilize particles in methods like the aptly named Particle Levelset method [EFFM02] of Enright et al. In the method they tie the signed distance function to a family of particles situated near the interface. The particles are used to correct the levelset from the inherent information loss that interpolation causes during levelset advection.

Traditionally Semi-Lagrangian simulation has depended on particles because managing topological changes in fluid simulation is an arduously difficult problem to solve. More recently, meshing technology by those like Brochu et al. [BB09] has advanced far enough that using Lagrangian interface tracking with triangle meshing has become feasible [WMFB11]. The main disadvantage of these methods is the necessity to invest time to develop and tune a mesh cleaning tool. Many methods have requirements that the tessellations they use must be manifold or delauney. Another common reason why a decent mesh cleaning is required is to make sure that the tessellated elements do not become too small or too large. Elements that are too small restrict the size of each timestep in the simulation, which reduces performance, while large elements can reduce the quality of the simulation.

3.2 Granular Simulation Techniques

Granular simulation has primarily been performed through purely particle-based methods, but recently there have been more attempts to simulate them as a continuum in order to handle scalability issues that arise from simulating too many particles.

3.2.1 Particle Granular Materials

The most obvious way to simulate dry granular materials is as a collection of rigid bodies. Although many granular materials have a variety of anisotrophy that varies from being simple ellipsoidal shapes like in rice grains or fairly complicateted geometries of "arbitrary" shapes such as is found in sand, a common assumption is that in aggregate they act like spheres. As such, the most common form of simulation of granular materials is as a familiy of rigid spheres, despite the actual geometry of what is being simulated [YHK08]. The work of Bell et al. [BYM05] allowed for the simulation of non-spherical particles, but they only allowed for objects represented as the union of spheres held together statically. From that perspective simulating dry granular materials the problem becomes simply a particular use case of rigid body simulation [KP06, Llo05], especially because the contact properties of spheres are so easy to implement. Recently there has been some interesting work by Smith et al [SKV+12], which is able to preserve spatial symmetries, kinetic energy, and momentum in large rigid body systems, which extends nicely to granular media. Through their work they were able to reproduce complex emergent patterns generated by vibrating granules in rectangular bins.

Although dry granular materials are well represented as spherical rigid bodies, cohesion is a difficult force to represent. There has been work such as that of Lenaerts and Dutré [LD09], which utilize SPH to simulate wet granular materials. They utilize SPH for both the fluid and the granular components of the simulation. They utilize their preceding work on porous flow using SPH [LAD08] to simulate the porous flow of fluid entering bodies of sand. Further work in simulating wet granular materials through SPH was explored by Ihmsen et al. [IWT13], though it does not discuss how to do porous advection. The work of Rungjiratananon et al. [RSKN08] uses SPH for the fluid simulation, but uses the Discrete

Element Method for simulating the forces between granules as a sort of mass-spring system generated by the adjacency of particles in each timestep.

The Material-point method[BBS00] is a method that evolved from FLIP.TODO: find earlier references.

Despite significant improvements on the field, solving for rigid body dynamics for large numbers of particles is an inherently intractable task. There have been several interesting approaches, however, such as the work of Zhu et al. [ZY10] which matches particles with a height field which allows for particles to be labeled as rolling, interface, or static. Through that separation they only need to worry about a narrow band of particles undergoing dynamics. The method is optimal for simulating sand being dropped or undergoing low magnitude forces, but because there is no simulation in the lower depths of material, the method is not amenable to other situations.

3.2.2 Continuum Granular Materials

The first explicit use of continuum methods for simulating granular materials was in the original graphics paper for FLIP by Zhu et al. [ZB05]. The method implements friction through a linear scaling of tangential velocities and clamping them to a non-negative value. They even support cohesion in their model, but mainly to remove some seepage issues that were a side effect of using such a simple friction model.

To solve for an even "stronger" form of incompressibility than what is used in FLIP, Narain et al. utilize the Unilateral Incompressibility Constraint [NGCL09] in an effort to guarantee that the constituent particles will not intersect with each other. Their application for that result was not for fluid simulation, but rather for simulating large numbers of independent agents such as crowds navigating through terrain as sophisticated as the interior of a building. They were able to handle large quantities of agents because, although the agents individually only respond to local stimuli, global flow issues are handled through a linear complementary problem, which enforces a density constraint for each cell in a regular grid. The constraint maintains an L^2 optimal set of forces such that no cell can have higher density of agents than that prescribed by the optimal sphere packing density.

By setting these particle-agents to advect in the same fashion as a FLIP fluid simulation, and by replacing the viscosity computation with a constrained energy minimization simulation of friction, Narain et al. were able to convincingly simulate granular materials as a continuum flow [NGL10]. This constrained energy minimization problem is solved with a quadratic programming problem, where the yield criterion between shearing strain and pressure formulated approximately through a set of constraint hyperplanes.

3.3 Porous Flow

Porous flow, also called Darcy flow, lacks the visual intrigue that vortices of smokes and splashing of water, but has serious consequences on the rheological behavior of materials, as in sand and paper. Although we have mentioned some methods that can support cohesive granular materials, and even wet ones, they do not contain a proper discussion of porous flow. Some of the work like Zhu et al. [ZB05] and Ihmnsen et al. [IWT13] simulate cohesive granular materials, but are silent on the fluid causing the fluid to be cohesive, while in the work by Rungjiratananon et al. [RSKN08] water particles are removed upon contact with granular particles to accumulate "wetness" on each granular particle. In that method,

the "wetness" of individual granular particles are propagated only when a particle becomes "over-wet", simulating that a granular particle only propagates wetness once it becomes saturated with water. They also assume that capillary forces are insignificant compared to gravity.

Within Graphics, the study of Darcy flow has not received much attention, only really being mentioned in the work of Lenaerts et al [LAD08] that derived everything through SPH. We found that the book by Bear [Bea72] is a very good resource for the general flow of the phenomenon, including some insights on the nature of capillary pressure and individual models for different pore topologies in porous media. Furthermore, the work by Hirani et al. [**?**] discuss how to solve the problem of Darcy flow using Discrete Exterior Calculus.

Chapter 4

Overview of Simulation Methods

In this chapter we provide a high level description of the core components of our method. The method combines two independent simulations: one for the fluid component and one for the granular material. We then connect the two simulations by mapping pertinent physical characteristics between the two simulations as parameters. Through the synchronization of these parameters in the respective simulations we obtain a coupling between the two simulations that is visually convincing.

In order to utilize these parameters in our simulations of granular materials and newtownian fluids we had to augment existing techniques for both the fluid and granular simulations, which we will soon discuss. This chapter will walk through the process of augmenting and combining simulation methods through a series of steps. We will begin by describing a novel method for computing a Hodge-Star from a levelset, which is a crucial component in everything that follows. From there, we will discuss a method for simulating a free surface fluid immersed in a porous body where porosity is variable over both time and space. Independently we will describe a method for simulating a cohesive granular material where the cohesion can change over time and space. Finally we will discuss how to combine the two techniques to provide a method for simulating the interactions between a fluid and granular body.

Before we enter the core of our work we will provide a light discussion on our choice of discretization.

4.1 Operator Choice

We utilize operators from exterior calculus, as described in A. The primary reason for utilizing this extra structure is that it makes implementation much simpler, while at the same time making the theory more general to other types of discretization. This ease of implementation is due to the small number of operators that need to be implemented and the simplicity of those individual operators, which makes it possible to create a full working system with less code to implement and debug. Furthermore, those operators are amenable to a wide variety of other discretization techniques beyond the one that we chose. This is because they only depend on having a cell complex with some reasonable concept of a dual mesh attached. This mathematical language is also dimension agnostic, which implies that the algorithms defined here have easily extensions to even higher dimensions with minimal effort, which accelerated our conversion from 2D to 3D code.

4.1.1 Cubical Complexes

Although the following theory is trivially generalized to other sorts of complexes, we will be only using cubical complexes. We choose to only discuss cubical complexes because although other structures, like simplicial complexes, are from a theoretical perspective simpler, we want to show that our method is not only equivalent to other existing grid-based methods and can even be easier to implement than them as well. We include some useful implementation details in Appendix B which defines the cubical structure somewhat rigorously and presents the simplifications that cubical complexes have over generic cell complexes.

4.2 Levelset Based Hodge Star

As discussed in the Appendix B the Hodge Star operates as a way to map between k-forms and $(n-k)$-forms and allows for the definition of an inner product on differential forms. For simplicity of implementation DEC uses the diagonal Hodge Star operator, a sort of mass lumping that guarantees that between a k-cochain and a $(n-k)$-cochain that the evaluation of a k-form and its respective $(n-k)$-form on the k-cochain and $(n-k)$-cochain produce the same value. In order to guarantee an isomorphism between k-cochains and $(n-k)$-cochains in the discrete setting, DEC introduces a dual mesh, for which the Hodge Star defines a correspondence between the primal mesh and its dual mesh. In general the functionalities available to the primal mesh are identical to those available to those of the dual mesh and the difference comes down to the topological and geometric features of the choice of primal and dual meshes.

For instance, the vertices of the dual mesh are usually defined by the circumcenters of simplices, which is created by attaching a vertex at the circumcenter of every n-cochain, a dual edge between dual vertices for which their corresponding primal n-cochain share a $(n-1)$-cochain, and so on. The above construction leads to a trivial computation for the discrete boundary operator of the dual co-chains:

$$\bar{\partial}_k^{k+1} = \left(\partial_{n-k-1}^{n-k}\right)^t$$

where $\bar{\partial}_k^{k+1}$ denotes the dual boundary operator and $\cdot^t$ represents the transpose operator. In later sections we will not distinguish between primal and dual operators because the choice of storage between the primal and dual mesh is generally arbitrary. The Hodge Star operator, however, is the main correspondence between the primal and dual meshes so we will still talk about primal and dual meshes for the remainder of this section.

The discrete Hodge Star maps a k-form ω^k sitting on k-cochain σ^k to a $(n-k)$-form $\bigstar\omega^k$ on $(n-k)$-cochain σ^{n-k}. We compute the value of the Hodge star by first recalling its definition:

$$\frac{1}{\|\sigma^k\|}\int_{\sigma^k}\omega^k = \frac{1}{\|\sigma^{n-k}\|}\int_{\sigma^{n-k}}\bigstar^k\omega^k.$$

By using the integrated quantities as the discrete value for the forms and by letting $\int_{\sigma^{n-k}}\bigstar^k\cdot = \bar{\bigstar}^k\int_{\sigma^{n-k}}\cdot$

we obtain the diagonal Hodge Star:

$$\begin{aligned}\frac{1}{\|\sigma^k\|}\int_{\sigma^k}\omega^k &= \frac{1}{\|\sigma^{n-k}\|}\int_{\sigma^{n-k}}\star^k\omega^k\\ \frac{1}{\|\sigma^k\|}\bar{\omega}^k &= \frac{1}{\|\sigma^{n-k}\|}\bar{\star}^k\bar{\omega}^k\\ \frac{\|\sigma^{n-k}\|}{\|\sigma^k\|}\bar{\omega}^k &= \star^k\bar{\omega}^k.\end{aligned}$$

Standard DEC is designed around having a static primal and dual meshes but we extend the concept of the diagonal Hodge star to describe more dynamic situations.

Our method operates by maintaining a static dual mesh (in this case a regular cubical complex) and extrapolating a dynamic primal mesh extrapolated from a levelset. A standard problem with levelsets is that they do not have any inherent information on the input geometry within a single cube, so the particular geometry inside of a cube is underdetermined. We therefore do not generally have enough knowledge to explicitly generate the primal mesh, and in fact take advantage of the fact that we have some unknown information to virtually pick the interior data that satisfies our wants. In particular we use the levelset to determine the Hodge star directly from the levelset using linear interpolation schemes.

As levelsets are scalar fields, it is natural to store the levelset on the primal or dual 0-forms, and for each of these types of levelsets there are different ways to extrapolate quantities.

4.2.0.1 Length of edges

For a levelset ϕ_p stored on the primal grid we can compute the length of the primal edge $e = v_i, v_j$ from the levelset directly. If $\phi_p \leqslant 0$ at both points then $vol(e) = \Delta x$ and if $\phi_p > 0$ on both then $vol(e) = 0$. WLOG, assume $\phi_p(v_i) \leqslant 0$ and $\phi_p(v_j) > 0$ and we can set:

$$vol(e) = \Delta x \frac{\phi_p(v_i)}{\phi_p(v_i) - \phi_p(v_j)}$$

which is, under the assumption that $\phi_p|_e$ is linear, Δx scaled by the fraction of the edge such that $\phi_p < 0$.

If the levelset ϕ_d is stored on the dual grid, the length of a primal edge requires a bit more effort. If we assume that levelset is bilinear within each dual cube we can average the values of ϕ_s at each of the dual vertices to generate a primal ϕ'_s, for which we can use the above procedure for generating primal volumes on primal grids.

The obvious analogues work for computing edge lengths on a dual mesh.

4.2.0.2 General Volumes

In general, for higher dimensioned edges there is more freedom in how one can compute volume from a levelset. As in the case of edge lengths, for dual elements we have to take averages to translate the primal levelset to a dual levelset. We assume that objects are piecewise linear and utilize the volumes of the triangles/tetrahedrons generated by evaluating a square/cube respectively using marching squares/cubes.

Once we have the volumes, we can generate the Hodge star operator between the primal and dual elements through dividing the primal and dual volumes and vice versa.

4.2.0.3 Dealing with Degeneracies

One complication with these volumes is that we may have degenerate elements (elements with 0 volume). When these entries are used to compute the Hodge star, the resulting operator is no longer an isomorphism between the primal and dual forms, which is a fundamental property of the Hodge Star. This, however, is not an issue, as the places where the Hodge star is degenerate are precisely the places where computation is unimportant. In our problems our only concern is to solve for pressure inside the fluid domain, which is definitely outside of the solid domain. The Hodge star on that restricted region is guaranteed to be nondegenerate because the levelset of a solid is always disjoint from the levelset of a fluid, and the levelset for a fluid is always inside the levelset of a fluid. By zeroing the Hodge star when we get zero or infinite volume ratios we obtain a projection operator on forms that projects into the restricted space.

This projection, along with some numerical considerations, will create some additional restrictions on how we set up the system. One important consideration is that we will not be projecting the boundary operator to the projected space, which will result in an underdetermined system unless we take special care in whether to store our quantities in the primal and dual meshes.

4.2.1 Pressure Projection

Of course the final goal of all of this theory is to produce a satisfactory system of equations to perform the standard pressure projection step in fluid simulation. By storing pressure p as a primal n-form and velocity u as a dual 1-form we we can obtain the amount of "pressure", we can decompose u by using the Hodge decomposition to obtain

$$\bar{u} + d \star p = u$$

where $\bar{u}$ is the summation of the two remaining terms from the Hodge decomposition: $\bar{u} = \delta\beta + \gamma$. Since $\delta\delta = 0$ and the fact that $\delta\gamma = 0$ for harmonic functions, we see that $\delta\bar{u} = 0$.

Because $\bar{u}$ is in the kernel of δ we can do the following:

$$\begin{aligned} \bar{u} + d \star p &= u \\ \delta\bar{u} + \delta d \star p &= \delta u \\ \star d \star d \star p &= \star d \star u \\ d \star d \star p &= d \star u. \end{aligned}$$

By storing the dual of pressure as $\bar{p}$ we get the weak Laplace-deRham operatorc̃iteabraham1988manifolds:

$$\begin{aligned} d \star d \star p &= d \star u. \\ d \star d\bar{p} &= d \star u. \\ \mathrm{L}\bar{p} &= d \star u. \end{aligned}$$

This produces a positive symmetric semidefinite linear system, which can solved efficiently using preconditioned conjugate gradient, with a modified incomplete Cholesky factorization as the preconditioner,

following [BBB07].

From there one can easily find a value $\bar{u}$ by

$$\bar{u} = u - d\bar{p}$$

which is the L^2 projection of u into the space of gradient free fields.

4.2.2 Free Surface Boundaries

We require one further addition to compensate for the force of pressure on the fluid boundary, which is taken care of by using the ghost fluid method mentioned in [EMF02]. With a free surface boundary the volume of fluid is not contained and the pressure from the air must be taken into account. The standard solution is to set a Neumann boundary constraint that the pressure at the boundary is 0, which is what we do. However, applying the boundary condition with levelsets is difficult because the interface is not precisely defined on a grid boundary, and so a method such as the ghost fluid method is necessary. The ghost fluid method uses Neumann constraints to extrapolate values of a scalar quantity right outside the boundary. This is particularly relevant in our case, where the quantity is pressure and the boundary value is always 0.

If we use a linear extrapolation of pressure from a cell-center p_i that is inside to generate the value of a cell-center outside p_j, such that $\phi(x_i) = -\theta$ we see that the values must satisfy

$$0 = (1+\theta)p_i - \theta p_j$$

which provides us with

$$p_j = \frac{1+\theta}{\theta} p_i.$$

If we look at the effect of the exterior derivative operator d on p_i, p_j, we see that on the face they share we get, up to sign dependent on orientation,

$$dp|_\Omega = p_j - p_i|_\Omega = \frac{1}{\theta} p_i = \frac{1}{vol(e_{ij})} = \bigstar^l|_\Omega p$$

where $\bigstar^l|_\Omega$ is the Hodge Star restricted to the boundary. If we take $d\bigstar^s$ of the operator to generate a full Laplacian, notice that by definition of the situation only the cell storing p_i will only have one side of the pertinent axis nonzero, so the $d\bigstar^s$ simply maps the the face value to the vertex cell-center. By noticing that we are purely adding terms to the system, the result of the linear extrapolation is the following system:

$$(\mathrm{L}^l + \bigstar^s \bigstar^l|_\Omega)\hat{p} = d \bigstar^l u,$$

It is easy to convince oneself that this is still a positive semidefinite symmetric system which is advantageous for numerical solvers.

4.3 Darcy Flow

To extend the above method to support flow in a porous medium, also called Darcy flow, there are two key insights. The first is that the porosity can be treated as a scaling of the permeability through a plane, which leads to a porosity-based Hodge Star. The second is traversing through a porous medium mutes the momentum of a porous material, and so the velocity from a previous step has no bearing on the velocity of the current timestep. In fact, the velocity of fluid in a porous medium is determined by velocity of the porous medium itself, the pressure effects to maintain incompressibility, and the force capillary pressure.

4.3.1 Porosity

In our simulation technique we use a scalar porosity function ρ across the whole domain, with the value 1 specifying that fluid can fully fill a region and 0 signifying that the body in that space is completely filled. By defining porosity as a scalar function across over our entire domain we obtain a representation that allows for variable porosity throughout both time and space. When $\rho = 1$, there are no obstructions to fluids flowing through that neighborhood, so any fluid in the region $\{x : \rho(x) = 1\}$ can be represented as a standard free surface fluid. However, when porosity is below some threshold we assume that the fluid in the region will be under a porous flow regime and will have the same velocity as the bulk material surrounding it.

4.3.2 Porosity Based Hodge Star

We model the porosity as the percentage of a volume that can be filled with fluid material. The amount of available volume is the sum of available volume between open air and the volume available in porous solid materials. We assume that all solid material in a region is porous with the same porosity ρ found in a grid. That is, the solid volume in a region of space V is

$$(1 - \rho)\text{vol}_{\phi^s}(V).$$

The total amount of fluid that can fill a volume is then defined by a scaling of the solid volume and the remaining available space:

$$\rho\text{vol}_{\phi^s}(V) + (\text{vol}(V) - \text{vol}_{\phi^s}(V)).$$

If we modify the Hodge Star operator to take into account this modified fluid volume we obtain a Hodge Star that represents the porous medium. The previous assumption could be removed by keeping track of the porosity of each constitutent object in a solid levelset and subtracting the solid volume filled by the materials appropriately. By using the above formulae we can define modified versions of the fluid Hodge Star operator, and apply the same pressure solving technique as before.

4.3.3 Capillary Forces

Our capillary forces are derived by the difference in potential between the wetted and dry phases of our granular material, in our case water and air.

There are various models that assume different microstructure geometries [**?**] but we have chosen to go with a simple $\sqrt[n]{\cdot^{n-1}}$ relationship to map volume to surface area due to its simplicity. Most

graphics fluid methods assume that air is really a vacuum, and we will continue with that except for when considering capillary potential, which depends on the existence of air. In particular we do not consider wind blowing on the sand, leaving this topic to future work.

We chose to represent this capillary potential Φ by setting the potential on air-surface cells to be 0 and on fluid-surface cells to be a function of the fluid density per cell, while taking into account the surface area of porous material available. By using the previously mentioned approximation between volume and surface area, and by taking into account that the fluid-solid interface cannot be more than the minimum surface area of the two, we obtain

$$\Phi = \sqrt[n]{\min(Vol_f(V), Vol_s(V))}^{(n-1)}.$$

Though this is an extremely crude approximation of the potential between the two areas, it is not the focus of this work. A correct model for the solid porous region would require analysis on a per-object basis and would likely exceed our needs for our phenomenological purposes. A more careful scheme is left for future work.

4.3.3.1 Maintaining Particle Density

A basic FLIP implementation has no mechanism for managing the density of fluid. This is very important when the porosity of a region changes such as when a sponge is squeezed. Although FLIP generally has satisfactory volume preservation properties, the particles in a FLIP simulation have a tendency to clump together. Once they do, they rarely ever separate, which causes volume loss. [NGCL09] utilized a lightweight algorithm to separate particles in each timestep in their UIC simulation, which we likewise apply in our FLIP simulation. The scheme applies symmetric pair-wise position modifications in an attempt to cheaply remove as much self-intersection as possible. Im a general flow of particles such a scheme fail in when multiple particles converged, so these corrections would not work. However, because fluid particles move in a relatively divergence-free fashion, the piecewise corrections become the sufficient nudge required to prevent the vast majority of intersections. In fact, we applied this pairwise scheme on a normal FLIP simulation and observed almost no particle self-intersection and significantly less volume loss without any seemingly different phenomenology.

This method of particle separation does not require both sets of particles to be of the same size, though one could imagine making the repulsion match the interacting particles' radii. Because the fluid simulation method described so far does not explicitly take into account the density change that occurs when a parcel of fluid goes into a porous material, the density of fluid inside a porous region could be equal to the density outside, which is not physically plausible. Because the porous material fills space, the fluid must diffuse and fill more volume at a continuum level. In order to compensate for the required density loss we change the radius of each fluid particle in order to give the particle a constant volume while immersed in a porous medium. Specifically, given a particle with air-based radius r in a porous medium of porosity ρ, we satisfy the equality

$$\frac{4}{3}\pi r^3 = \rho\frac{4}{3}\pi\bar{r}^3$$

with the new radius being $\bar{r} = \frac{1}{\sqrt[3]{\rho}}r$. In higher dimensions, of course, the radius would be scaled by $\frac{1}{\sqrt[n]{\rho}}$.

We utilized spatial hashing in order to accelerate the performance of the pointwise comparisons. By

creating cell-center based storage grid that could hold vectors of grids we stored particles on these grids and performed the pointwise comparisons between every particle in a single cell with all of the adjacent cells.

4.4 Cohesive Granular Materials

The rheological behavior of wet granular materials is a generalization of the behavior of granular materials that stick together. We therefore first discuss how to simulate a granular material that has cohesive forces sticking the particles together. Although cohesion holds particles together at a per-particle-pair isotrophic force, at a continuum level the effect of cohesion on a bulk of granular material results in both isotropic and deviatoric strains.

4.4.1 Isotropic Strain

The theory behind how to deal with isotropic strain is quite simple: each particle induces an attractive force to every other particle to which it is connected. In practice, we utilize the most rudimentary method possible: we introduce attractive forces between two particles in order to introduce particle level cohesion. We generate this particle-wise force during the collision detection step in order to take advantage of the existing distance comparisons. We add cohesive forces to particles a of distance less than $1.1 * radius$ through a radial cube spline function once collisions have been dealt with.

The simple, explicit scheme described above is too naive to support more delicate overhanging behaviors, but it appears to be sufficient for small scale clumping and for preventing the boundary of wet material from eroding. The use of a more sophisticated method would be prohibitive at the scales in which we are interested, but would definitely be interesting further work.

4.4.2 Deviatoric Strain

The deviatoric forces are a direct result of the complex structure generated by a network of cohesive particles pulling on each other. This results in a force to resist shearing, which increases the amount of shearing strain required to make a bulk yield. This is why the yield criterion utilizes cohesion as an added scalar factor to the right hand side of the yield criterion:

$$\|s\| \leqslant \sqrt{3}\alpha p + c.$$

4.4.2.1 Discretization

Since the deviatoric effects of cohesion are primarily an artifact of homogenization of a bulk of granular material, we can only discretize it at a continuous level. We therefore only deal with these effects at the continuum level of our representation.

The approach we take at the continuum level is through an extension of the original dry granular material method of Narain et al., but with some further modifications to support the effects of cohesion.

With cohesion, the system of discrete formulation we obtain is

$$s^T A_2 s + b_2^T s \geqslant 0 \tag{4.1}$$

$$(s_{\max} + c) \geqslant s \geqslant -(s_{\max} + c) \tag{4.2}$$

$$\tag{4.3}$$

where A_2 and b_2 are defined as they were before at 2.8 and 2.9 respectively.

As used by Narain et al., we solve this system of equations by using a customized implementation of Modified Proportions with Reduced Gradient Projections, which we will discuss in more detail in the technical details section in order to handle the above linear complimentary problem.

One artifact that appears is that any cell that has nonzero cohesion is connected in this linear system. Cohesive cells with few particles act as if they are filled with some cohesive aether and prevent actual particles from moving. Because of this issue some care must be taken to reduce the amount of cohesion in order to prevent cohesive forces from being generated by cells with few particles. We choose to not include cohesion in cells where the particle density is below some threshold that we manually tuned.

4.4.3 Using Levelset Hodge Star for UIC

Similar to the original paper on the Unilateral Incompessibility Condition [NGCL09] we switch from using the Laplacian $d^T d$ to using the weak Laplace-deRham operator $d^T \star d$ to take into account the amount of solid available in a region. In the original UIC discussion the population density is considered, but we utilize our existing code for computing the Levelset Hodge Star to approximate the granular density in a cell.

4.5 Wet Granular Materials

The final step is to merge our above methods for free surface fluid simulation with porous materials and cohesive granular materials into a single simulation method for a coupled simulation of fluid and granular materials.

As we have mentioned before, we create this coupling by changing the porosity of space in the fluid simulation and by changing the cohesion of particles in the granular simulation. We do this through a fairly simple connection at the continuum level and through a bit of care at the particle level.

Porosity is particularly easy to define; it is simply the fraction of volume, per unit volume, that is free:

$$\rho = \frac{\text{vol}(V) - \text{vol}_{sand}(V)}{\text{vol}(V)}.$$

On the other hand, the cohesion forces introduced by the fluid into the sand simulation is something we need to model. In reality computing the cohesion requires analysis on the aggregate behavior of particles being held together with water without any consideration of contact geometry or material properties. For simplicity, we utilize a simple linear function of the fluid volume fraction in a cell as well:

$$\phi = \frac{\text{vol}(V) - \text{vol}_{fluid}(V)}{\text{vol}(V)}.$$

4.5.1 Ordering Operations

There are two crucial components of each simulation method which require careful ordering of operations. The first is that we need to pass the coupled quantities between the two simulations, and the second is that particle advection needs to be performed with care so the fluid in a mass of granular material moves with the granular material.

We chose to follow a simple approach for the passing of coupled quantities: we generate the porosity and cohesion grids at the very beginning of each timestep. This choice seems reasonable to us because the various coupled quantities only depend on particle densities, and those densities only change during advection, which is at the end of each timestep. Thus throughout the evaluation of a timestep, until the advection at the end, the coupled quantities are consistent with the simulations from which they are derived.

4.5.2 Advecting quantities

For advection we look at the momentum stored in each of the velocity grids to get a velocity field for the whole system. In general, we maintain separate velocity grids for each simulation, but for only the advection part of FLIP/PIC we pull velocities from this merged grid:

$$u = \frac{m_{fluid}u_{fluid} + m_{sand}u_{sand}}{m_{fluid} + m_{sand}}.$$

For regions where there is only granular material or only fluid, the velocity maintains the same value as the respective velocity field, and in merged regions we obtain a unified velocity field.

4.6 Summary

We began this chapter by defining the Levelset Hodge Star, which allowed us to reinterpret the pressure solve from of Batty et al.[BBB07] and the first order ghost fluid method [EFFM02] in the language of Discrete Exterior Calculus. This reformulation allowed us to then generalize the Levelset Hodge Star to support porosity, which led to a porosity-aware pressure solve. We then developed a model for capillary pressure and a method for obtaining reasonable particle densities in porous materials. With those three components we were able to develop a method for simulating fluid flow in porous materials.

We then developed local and global modifications to the granular simulation method of Narain et al. [NGL10] to support cohesion. Finally, we combined the two simulation methods to create a technique for simulating the interactions between a fluid and a granular material.

Chapter 5

Technical Details

5.1 Introduction

In our opinion the most challenging components of the techniques we discuss in this thesis are the numerical solvers required for solving systems efficiently. The systems we solve all utilize banded, symmetric, positive semidefinite problems. We know that to be true because all of our matrices are generated by $A^T A$ or $A^T DA$ where D is a diagonal matrix, where A is a sparse banded matrix. Therefore we have a perfect setup for using Krylov subspace methods - namely the conjugate gradient method. A great reference for both solving linear systems and for doing conjugate gradient is Robert Bridson's SIGGRAPH 2007 notes [BMF07] so we will not go over the details here. In particular, it discusses how to use MIC0 preconditioned conjugate gradient which is one of the best choices for solving linear systems composed of the types of matrices we encounter.

For our systems that are quadratic programming or linear complementary problems, however, standard conjugate gradient doesn't directly work because of the boundary constraints. Therefore we use a form of conjugate gradient that can handle linear inequality constraints, namely MPRGP (Modified Proportioning with Reduced Gradient Projections). If the reader needs a refresher on quadratic programming or linear complementary problems please refer to Appendix C.

It's worth noticing that in the cases where our systems aren't positive definite we can use the minimal-norm solutions to the systems of equations for two reasons. First, because we only use the derivatives of our solutions, adding a constant to the coefficients doesn't change the results.

Second, because our levelset Hodge star is applied on the solutions, extraneous information from entries represnting data outside of the levelset are zerod out and ignored in the min-norm solution. Because MPRGP is a fairly recent algorithm [DS05] and we need to modify it, we will discuss it in detail for completeness.

5.2 Modified MPRGP Quadratic Programming Solver

Like [NGCL09] we utilize a version of MPRGP with modifications for handling our constraint manifolds. For satisfying the Unilateral Incompressibility Constraint we can utilize the algorithm without modifications, because the base solver requires a constraint manifold of the form $\{x = (x_i) | x \in \mathbb{R}^n, x_i \geqslant \ell_i\}$. However, for solving friction, the constraint manifold is $\{x = (x_i) | x \in \mathbb{R}^n, |x_i| \leqslant \ell_i\}$. We will now provide

some background knowledge for solving these sorts of systems.

5.2.1 Convergence Criterion

Having a good stop criterion for LCP and mLCP problems can be tricky because of the complimentary condition. For LCP residuals are also not useful because the constraints will usually force the solution to be far away from where the residual goes to 0. A particularly nice and easy error metric for a normal LCP system is given by

$$E = \sum_i (|z_i w_i| - \min(z_i, 0) - \min(w_i, 0))$$

where our current solution z and residual w are punished for negative values and also for being far from complimentary.

For mLCP we have a more sophisticated stop condition because we need to confirm that the residuals $\rho_a^k, \rho_b^k, \rho_c^k$ go to 0 where they are defined by

$$\begin{aligned} \rho_a^k &= \|Au^k + Cv^k + a\| \\ \rho_b^k &= \min_i\{v_i, (C^T u^k + Bv^k - w^k + b)_i\} \\ \rho_c^k &= \max_i\{0, -(C^T u^k + Bv^k - w^k + b)_i\} \end{aligned}$$

This leads to an error function

$$E = \max\left(\frac{\rho_a^k}{1 + \|a\|}\frac{\rho_b^k}{1 + \|b\|}\frac{\rho_c^k}{1 + \|b\|^2}\right)$$

We will use this criterion as our stopping criterion in the algorithm defined below.

5.2.2 Projection Methods

One common technique for handling the nonlinearity of LCP problems, which is how MPRGP operates, is to explore specific submanifolds of the entire constraint manifold. The usual restriction is that, given that the constraint manifold is a set of axis-aligned planes, to only modify a subset of the variables at a time. The set of variables that are currently active is called the active set.

5.2.3 Notation

Before we begin, with the details of the algorithm, we will first introduce some notation that will be used significantly in the algorithm. The manifold solutions that satisfies our constraints is defined as Ω. Let g be the current descent direction and let β and ϕ be vector functions defined as follows:

$$\begin{aligned} \phi_i(x) &= g_i(x) \forall i \in \mathcal{F}(x) & \phi_i(x) &= 0 \forall i \in \mathcal{A}(x) \\ \beta_i(x) &= 0 \forall i \in \mathcal{F}(x) & \beta_i(x) &= \min\{g_i, 0\} \forall i \in \mathcal{A}(x). \end{aligned}$$

where

- ϕ,the free gradient operator, describes the proportion of g that is not in a constrained dimension.

- β, the chopped gradient operator, describes the proportion of g that goes in a constrained dimension, though modified as to not break the constraint.

Together they form $\nu(x) = \phi(x) + \beta(x)$, which is 0 precisely when the KKT conditions are met.

5.2.4 Steps

The algorithm works by picking one of three step choices each iteration: a conjugate gradient step, an expansion step, or a proportioning step. The conjugate gradient step is precisely the standard conjugate gradient step, while the other two steps are corrections to take into account the constraints. The choice of step type in a given iteration is made by determining the relative magnitude of the current descent direction that breaks through Ω through.

The simpler of the two correctional steps is the expansion step. It's the reaction to when conjugate gradient is about to break a constraint: it moves the current solution in the same direction conjugate gradient would normlly go, but stops right at the constraint boundary. It's derived quite simply:

If we let P_Ω be a projection operator back onto Ω and $\bar{\alpha}$ some minimal step size parameter set in the interval $(0, \|M^{-1}\|]$ we set the expansion step to be

$$x^{k+1} = P_\Omega(x^k - \bar{\alpha}\phi(x^k)).$$

This projection operator can be made more explicit by defining a reduced free gradient operator $\bar{\phi}$ which will set x^{k+1} to activate the constraint in the step. It's defined by:

$$\bar{\phi}(x) = \min\{(x_i - l_i)/\bar{\alpha}, \phi_i\}$$

which, because $x^k \in \Omega$, allows us to write the projection operator in terms of

$$x^{k+1} = P_\Omega(x^k - \bar{\alpha}g(x)) = x^k - \bar{\alpha}(\bar{\phi}(x^k) + \beta(x^k)).$$

The logic for whether to use the proportioning step is checked before the conjugate gradient step length and is used to push away from useless active constraints as much as possible . This is done with the inequality

$$\|\beta(x)\|^2 \leqslant \Gamma^2 \bar{\phi}(x)^T \phi(x)$$

with a parameter Γ in $(0, 1]$ to determine the proportion of ϕ the gradient is allowed to face into Ω. If the proportioning step is chosen, it does one round of conjugate gradient using β as the gradient for that step.

5.2.5 Performance

Conjugate gradient, as a Krylov subspace method, depends on iterating through a sequence of matrix multiplies off of an initial vector v $\{A^k v\}$. This sequence of matrix products conflicts with projection operators because modifying only the active set destroys information of the subspace explored by the multiplication, thus ruining the convergence properties of the method.

The way that MPRGP deals with that fundamental issue with conjugate gradient is that it only uses a projection when the algorithm things that there will be sufficient gains by doing so. In our experience

Algorithm 1 Modified Proportioning witih Reduced Gradient Projections

Set $r \leftarrow Ax - b, p \leftarrow \phi(x)$
while $\nu(x) > \epsilon$ **do**
  if $\|\beta(x)\|^2 \leqslant \Gamma^2 \bar{\phi}(x)^T \phi(x)$ **then**
    $\alpha_{cg} \leftarrow r^T p / p^T A p$
    $\alpha_f \leftarrow \max\{\alpha : x - \alpha p \in \Omega\}$
    if $\alpha_{cg} \leqslant \alpha_f$ **then**
      Conjugate Gradient Step
      $x \leftarrow x - \alpha_{cg} p$
      $r \leftarrow \phi(x) - \alpha_{cg} A p$
      $p \leftarrow \phi(x)$
    else
      Expansion Step
      $x \leftarrow x - \alpha_f p$
      $r \leftarrow r - \alpha_f A p$
      $x \leftarrow P_\Omega(x - \bar{\alpha}\phi(x))$
      $r \leftarrow Ax - b$
      $p \leftarrow \phi(x)$
    end if
  else
    Proportioning Step
    $d \leftarrow \beta(x)$
    $\alpha_{cg} \leftarrow r^T d / d^T A d$
    $x \leftarrow x - \alpha_{cg} d$
    $r \leftarrow r - \alpha_{cg} A d$
    $p \leftarrow \phi(x)$
  end if
end while

this doesn't happen very frequently, mostly as the algorithm nears convergence on a solution. That sort of experimental evidence led us to try applying MIC(0) preconditioning on the conjugate gradient steps, which was effective.

Chapter 6

Results

In this section we will briefly describe our implementation and then continue to discuss some visual results of our algorithm.

6.1 Implementation Details

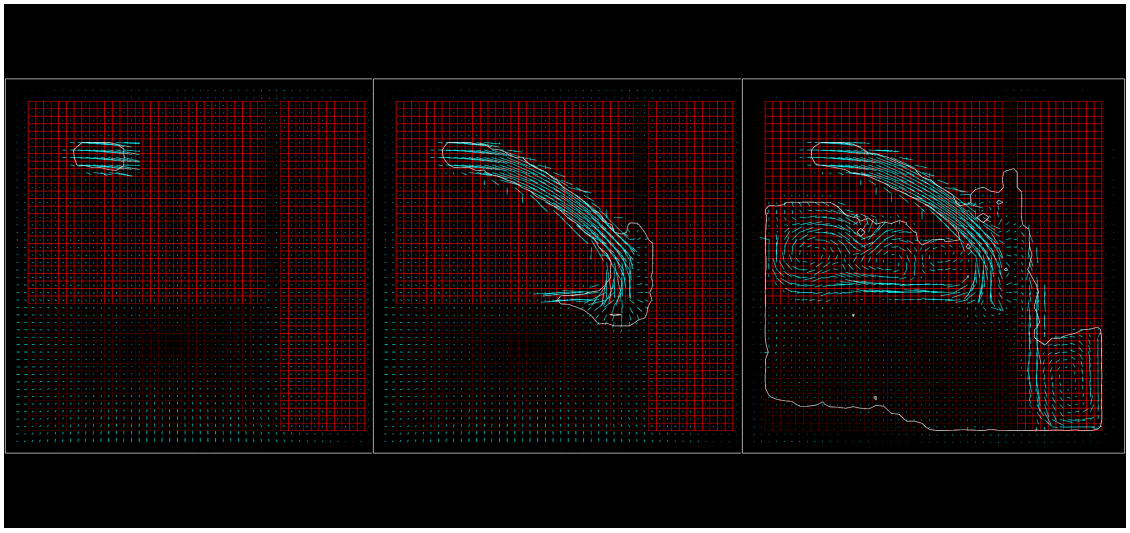
We implemented our method using C++ using Eigen as our matrix and vector storage mechanisms. We utilized C++ templates rather heavily in order to define the various grid types by the dimension of the forms that they held as well as to define and store the various differential operators. In general we found that template metaprogramming significantly improved the debugging experience because compilation errors would report bugs expediently rather than at runtime. This was especially convenient for storing quantities like the offsets used between different grids for their origins and dimensions.

We implemented our own linear and quadratic solver libraries, though our MIC0 cholesky factorizer was a translated version of Robert Bridson's code around Eigen's sparse matrix implementation. For generating scenes we also utilized Bridson's fast poisson-disk sampling scheme [Bri07].

6.2 Porous Results

Our method for simulating porous fluid can be seen in the sequence of images Figure 6.1, which displays several different phenomena related to porous flow. On the top left there is a circular fluid source, which we will call the emitter, that emits fluid to the right. To the right of the fluid source there is a porous wall that acts as a barrier that limits the flow of fluid from the left void to the void. Fluid still, however, does seep from the porous wall as the wall is saturated with fluid particles (Figure 6.1c).

The seeping action is particularly visible at the point where the emitter shoots water into the wall. Some of the seepage is caused by the fluid pushing itself into the porous body, as can be seen in Figure 6.1b: the velocity field inside the porous region indicates that the fluid is trying to push through the porous region. This phenomenon is caused by the porous pressure that we compute. The computed pressure makes penetration into the porous medium more difficult for the fluid than flowing back into the voids, so although some fluid is being pushed into the porous areas, splashing occurs and fluid flows parallel to the porous body. This splashing behavior is already visible in Figure 6.1b as the fluid begins to rush to the left after hitting the wall in a clockwise fashion even though the area is not yet saturated.



(a) (b) (c)

Figure 6.1: Fluid being spilled into the junction between a porous block and a porous wall. The redness of the background grid implies the porosity, the white boundary defines the fluid boundary, and the turquoise lines label the velocity field

Furthermore, in Figure 6.1c both voids have vortices that avoid the porous region because of how the pressure solve restricts the fluid from entering that region.

The effectiveness of using pairwise particle separation in standard FLIP fluid simulations can be seen in Figure 6.2. The fluid particles have very little self-interesction except at the boundary, which is a difficult region to prevent intersection because the particles have less movement options. Intersecting particles on straight boundaries can only move along those boundaries, which means that there is only a one-dimensional axis for which these particles can move. These sorts of density-based issues are much more noticeable in the 2D case than in the 3D case, where in general a larger proportion of particles will have more than one axis of available movement. Our method of modifying the particle radii according to the region's porosity is also visible in Figure 6.2: the particles in the porous areas have an appropriately lower density than in the voids.

6.2.1 2D Rendering Artifacts

Figure 6.2 illuminates two types of rendering artifacts that cause issues in our 2D boundary rendering that are easily fixed. These artifacts are the duals of one another: sometimes particles disappear and sometimes voids appear in fluid bodies. In order to avoid clutter when rendering sand particles we utilize a boundary based rendering approach. It is created by running the marching squares algorithm on the levelset we use for computing the Levelset Hodge Star. Because the grid is purposely coarser than the particles, sometimes the particles travel in paths that make them invisible to the marching sqaures algorithm. The algorithm depends on sign flips on edges to determine vertices and edges to render, so because there are no sign flips nothing is rendered. The inverse issue is that sometimes there are not enough particles in a sampling location, which then causes a void to be rendered although no effects like cavitation are being simulated.

The solution to these artifacts would be to increase the resolution of the levelset generated for

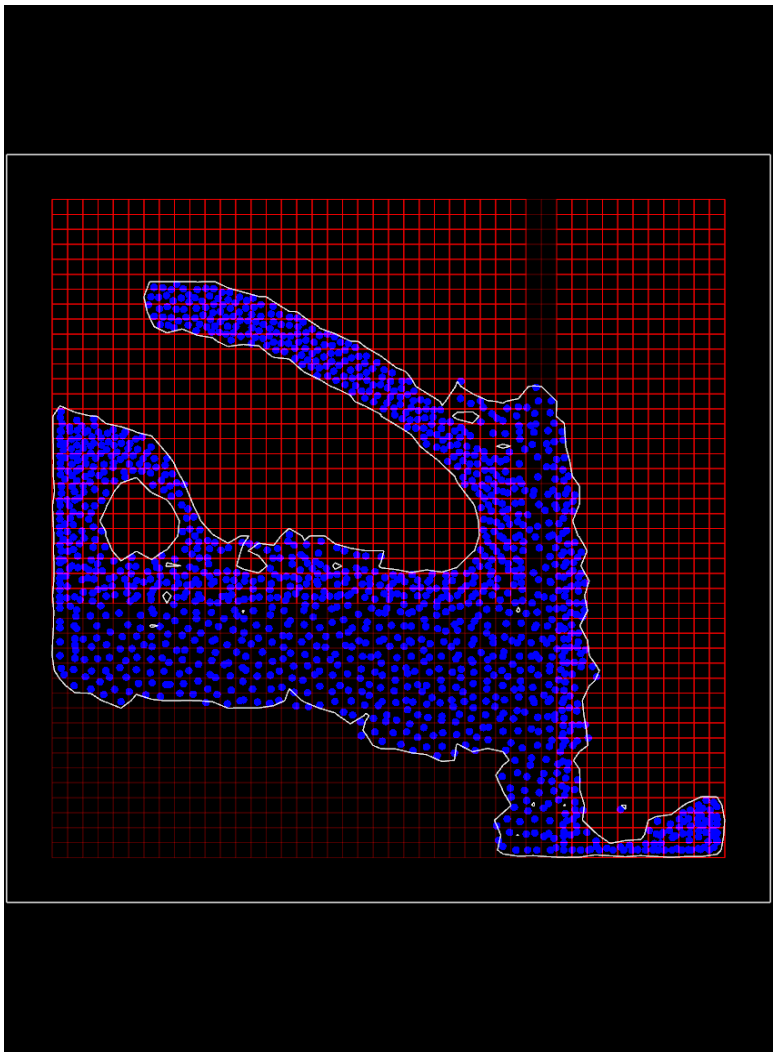
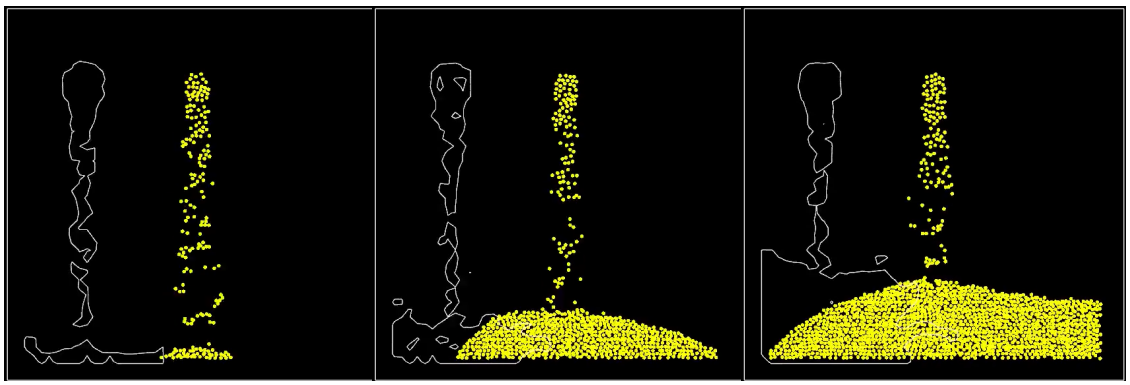


Figure 6.2: Figure of fluid particles rendered in the same simulation as Figure 6.1



(a) (b) (c)

Figure 6.3: Water and sand being emitted in near proximity

rendering, which is a trivial operation to do but we did not think it would be necessary to prove the effectiveness of our method for the purposes of this thesis.

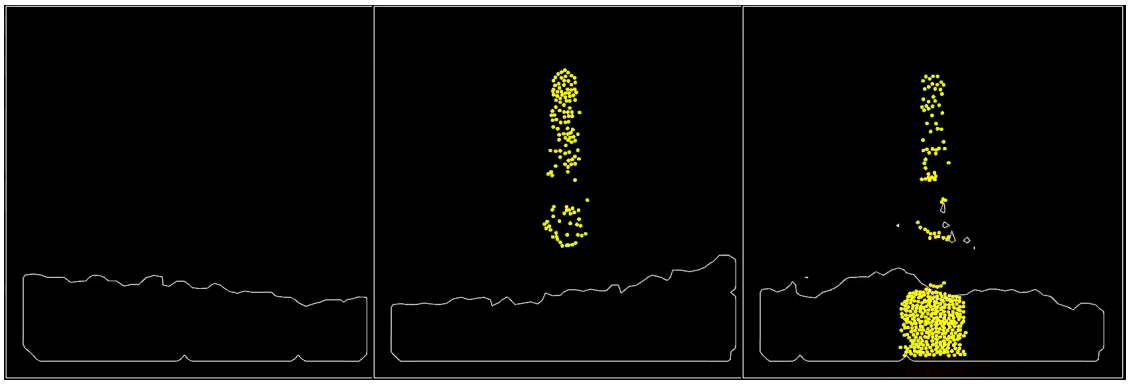
6.3 Combined Fluid and Granular Results

Our method successfully increases the steepness of sand piles and increases sand's resistance to movement. We focus on the steepness of piles generated because that is a useful characteristic for determining the effectiveness of cohesion. As the cohesion of sand particles increases, sand becomes capable of achieving steeper angles because particles hold each other together. It's this effect that makes the sort of overhanging behavior required to make sandcastles possible. In fact we are capable of producing a small amount of overhanging behavior.

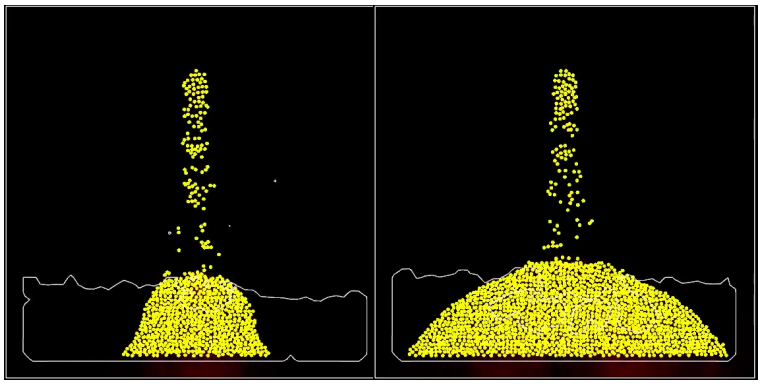
Figure 6.3 shows a scene where sand dropped from the center of a scene and water is dropped to its left. The cohesive effects of the water are clearly visible by Figure 6.3c as the sand that dropped forms a significantly more stable pile on the left side than on the right side. Because we wanted to generate higher piles we placed our emitters fairly high up, which caused impulses onto the sand pile. These impulses reduced the steepness of the piles generated. Furthermore, we had difficulty creating a constant stream of sand because particles coming out of the emitter tended to interact with other particles, causing the particles to come out in a spray.

In order to see the effects of sand falling into an existing cohesive region, we dropped sand into an existing pool of water (see Figure 6.4). In this simulation we were able to see that for very light, completely submerged piles, we were able to obtain very steep hills. The initial impact passed the momentum accumulated by the falling sand particles to shoot fluid particles into the air as can be seen in Figure 6.4e. Once the sand had risen above the water it once again formed a shallow pile and the submerged portion obtained a more shallow slope as well.

After we had run the simulation behind Figure 6.4 for a while tried turning back on the water emitter, which resulted in what we show in Figure 6.5. The configuration became reminescent of the porous scene for Figure 6.1 as the sand began partitioning the water into two parts. Some of the momentum from the fluid seemed to pass into the sand and the pile began leaning away from the water and some overhanging behavior emerged.

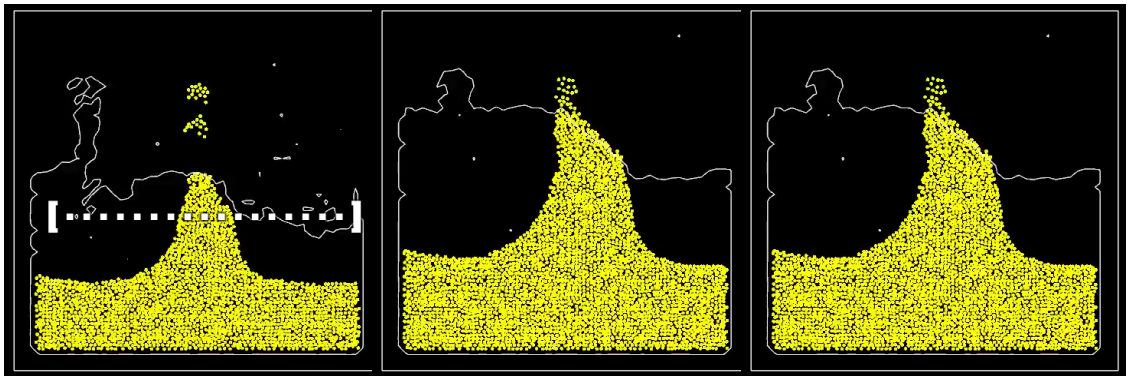


(a) (b) (c)



(d) (e)

Figure 6.4: Sand being dropped into water.



(a) (b) (c)

Figure 6.5: Water and sand being emitted in near proximity

Chapter 7

Concluding Remarks

Before we conclude this document, we shall first discuss some potential areas of future research for the methods we have described in this document.

7.1 Future Work

The most blatant omission of this work is that it doesn't have a working 3D counterpart, which we will look into in the near future. Beyond that there are still several issues with the method that need to be taken care. Many of the modeling assumptions in this paper were rudimentary and linear operators were chosen instead of measured quantities that correspond to actual material properties. Some of these modeling assumptions include the effect of wetness on cohesion could be done without changing the performance of the system because they would simply be changes to different grids.

Our method is not necessarily limited to only ball-like shapes and it would be interesting to see whether the simulation could be extended to more complicated or even heterogeneous particle shapes. That sort of exploration, however, would require some improvement on the stability of the system.

7.1.1 Stability Issues

We experienced a significant amount of stability issues with our sand simulation method. This was because particles bouncing into each other would significantly alter the velocity field in an area, which caused widespread vibrations that didn't stabilize. Using an alternative medium for connecting the particles to the grid like PIC instead of FLIP might have fixed some of those issues, but it would have made the dry sand lump together a bit more than desired. However, PIC would have been an excellent choice for advecting sand that was cohesive enough. Since the bulk of this work was done the material point method has gained some popularity in graphics [SSC+13] and would be appropriate for replacing the current granular simulation method.

7.2 Conclusion

In this work we have presented several novel contributions to the area of physically-based simulation in computer graphics. In order to achieve this result we have delved into the theory behind existing fluid simulation results for by reinterpreting their operators in terms of Discrete Exterior Calculus. Using

this reinterpretation we have extended those results to support porous fluid flow without much in the algorithms. We have also extended a dry granular material technique to support cohesion between particles at multiple scales. Finally we have prescribed a method for the simulation of interactions between fluid and granular media by utilizing the augmented fluid and granular media simulation methods we created.

We have only touched the begining of this area, as there are many ways for which this work could be improved. In particular stability of the granular simulation still is an issue, as is the fact that we must define a cutoff for which cells it is reasonable to label as cohesive. Furthermore the phenomology of interacting continuum materials like sand and air to create sand dunes is a poorly explored area. We hope to see further work published on novel physical phenomena, from both ourselves and others in the future.

Appendix A

Calculus of Differential Forms

Differential forms provide a graded algebra of the family of differential operators. The different grades of this algebra provide represent different types of fields suchas scalar fields, vector fields, tensor fields, fluxes, and metrics through a unified representation. Through operators like the Hodge star and exterior derivative, which operate on differential forms, we can reproduce many of the standard operators from vector calculus.

We will begin with defining differential forms in a familiar context before we provide a more rigorous definition. After that we will define some standard operators on differential forms and finally discuss the tools necessary to do integration, which is necessary for the discrete setting.

A.1 Ordinary Integration

Let us begin this discussion by first thinking about the components of a standard integral. From a traditional standpoint we are integrating over a subset of $\mathbb{R}^n$ like $[0,1]^n$ and will see a term like

$$\int_{[0,1]^n} f(x_1, \ldots, x_n) dx_1 dx_2 \cdots dx_n$$

where the dx_i are symbolic quantities that direct us of the axes for which we must integrate. For a more general manifold Ω this sort of coordinate-based representation might not feasable. What we want is a coordinate-free representation. One such representation is called differential forms. In this context we consider the dx_i as *differential one-forms*.

A.2 Tangent Bundles on Manifolds

Before we dig into the details of what differential forms are, let us first remember some elementary details about manifolds and their tangent bundles. We do this because we are now trying to integrate on *manifolds* which are a a sort of generalization of Euclidean space: a manifold is a collection of patches that are locally Euclidian, so when we need to do local analysis we can transport ourselves to a Euclidian space. We need the tangent bundle because that's where the things that we want to integrate live.

Consider what a vector v is in an n-dimensional space like $\mathbb{R}^n$, disregarding any manifold business for now. It's not a quantity in the space, but rather a quantity of a particular point. v sticks out from

a point p in a direction d, which means that it is really a equivalent to a tuple (p, d). Returning to manifolds, call the space of point/direction tuples the tangent bundle, or TM for a manifold M. For each point $p \in M$ we have a linear space of tangent vectors denoted T_pM. A vector at a point p is an element of the tangent space T_pM. The space of vector fields over M is the space of sections over TM, that is, the family of functions $V : M \to TM$ such that for any $x \in M, V(x) \in T_xM$.

As we have said earlier, manifolds are objects that are locally Euclidean. That means that we can pick one coordinate chart $\phi : \mathbb{R}^n \to M$ for a neighborhood on the manifold M to correspond between that neighborhood of M and a normal Euclidean space. For some orthogonal basis $\{e^i\}_{i=1\cdots n}$ we can obtain a basis for vectors in that neighborhood $\{\frac{\partial}{\partial x^j}\}_{j=1\cdots n}$ by evaluating the tangent map $D\phi : T\mathbb{R}^n \to TM$, which is defined by

$$D\phi_v(x) = \frac{d}{dt} f(x + tv)|_{t=0} = \sum_j \frac{\partial \phi}{\partial x^j}(x) v^j$$

at every point using the basis vectors.

The definition of a manifold provides a gluing map between coordinate charts taht allows for us to write any vector $\mathbf{X}_p \in T_pM$ as

$$\mathbf{X}_p = \sum_j X^j \left.\frac{\partial}{\partial x^j}\right]_p$$

This formulation of a vector on a manifold has a secondary notion that any vector is also a differential operator that we can use on functions.

In a Euclidean space the tangent bundle almost leads us to the space of affine transformations which discerns points and directions. The direction doesn't need a designated point because there's an implicit translation of the vector at the origin to every other point in space. On a manifold that implicit translation operator isn't available and we have to build it using an affine connection like the Levi-Cevita connection to obtain parallel transport, which is a sort of translation of vectors at one point to vectors at another. With the connection we can compare vectors in T_pM with vectors in T_qM, so in a Euclidian space we can endow the structure of an affine space by applying a Levi-Cevita connection and quotienting the the space TM by vectors that can be parallel transported to one another.

A.3 Differential Forms

Now consider the space of linear functionals that take elements from T_pM and map them to $\mathbb{R}$. This is called the dual space of T_pM and is usually denoted by T_p^*M and called the co-tangent space. The collection of all co-tangent spaces is the co-tangent bundle T^*M.

The natural pairing between vectors v and dual vectors $\hat{v}$ is denoted by

$$\langle v, \hat{v} \rangle.$$

The space of differential zero-forms is on a manifold is simply the space of scalar functions f on the manifold. When one takes the exterior differential operator d on a f one obtains a dual vector field. That is, for each point $p \in M, x \in T_pM$, $df_x(p) = Df_p(x)$ evaluates to a scalar value $\in \mathbb{R}$. This is precisely a section of the co-tangent bundle, and so differential one-forms are identified with T^*M. Just like we had the basis $\{\frac{\partial}{\partial dx^i}\}$ for vectors, we can obtain a basis for the dual vectors by identifying the vector dx_1 with the basis vector such that $\langle \frac{\partial}{\partial x^i}, d_i \rangle = \delta_{ij}$ where δ_{ij} is the Kronecker delta function. This produces

the family of differential one-forms

$$\Omega^1(M) = \{\omega = \sum_i \omega^i dx_i\}.$$

We can then read the differentiation of a zero-form f as

$$df = \sum \partial^j dx_j.$$

By using the wedge product we can extend the one-forms $dx_1, dx_2 \cdots dx_n$ to obtain differential two-forms

$$\Omega^2(M) = \{\omega = \sum_i \sum_j \omega^{ij} dx_i \wedge dx_j\}.$$

By continuing this process one can obtain differential forms all the way up to $\Omega^n(M)$, which is a one dimensional space comprised of scalar multiples of the determinant.

It's worth noting that $\dim\Omega^k(M) = \dim\Omega^{n-k}(M)$.

Intuitively, differential forms correspond to some common quantities. Though we won't elucidate the explanations too much quite yet, here are some some of the common quantities associated with the forms in $\mathbb{R}^3$

$\Omega^0(\mathbb{R}^3)$	$\Leftrightarrow$	Scalar functions
$\Omega^1(\mathbb{R}^3)$	$\Leftrightarrow$	Vector functions or velocity fields
$\Omega^2(\mathbb{R}^3)$	$\Leftrightarrow$	Vector functions or Flux
$\Omega^3(\mathbb{R}^3)$	$\Leftrightarrow$	Scalar functions or Volume

A.4 Differentiating Differential Forms

The exterior derivative d allows for one to jump from differential k-forms to differential k+1-forms through

$$d(f dx_1 \wedge dx_2 \cdots \wedge dx_k) = df \wedge dx_1 \wedge dx_2 \cdots \wedge dx_k.$$

For a finite dimensional space one can draw a diagram

$$\Omega_0 \rightarrow_{d_0} \Omega_1 \rightarrow_{d_1} \Omega_2 \rightarrow_{d_2} \cdots \Omega_n$$

which is usually called the deRham complex, which is a graded algebra. In the above diagram we have used d_i to denote the particular exterior derivatives from $\Omega_i \rightarrow \Omega_{i+1}$ and so far we are only able to differentiate to the right.

The exterior derivative corresponds to some common quantities in when M is $\mathbb{R}^3$:

$$\begin{aligned} \nabla f &= d_0 f \\ \nabla \times u &= d_1 u \\ \nabla \cdot u &= d_2 u \end{aligned}$$

where f is a scalar function (identified with a 0-form) and u is a vector field (identified with a 1-form

and 2-form at the appropriate times).

A.5 The Hodge Star

As we have just implicitly mentioned, 1-forms and 2-forms can both correspond to vector fields, which is a particular feature of being in $\mathbb{R}^3$. As we listed even earlier, there is a connection between 0-forms and 3-forms in $\mathbb{R}^3$ as well, all thanks to the Hodge star, which has the prototype

$$\bigstar_k : \Omega^k(M) \to \Omega^{n-k}(M).$$

The Hodge star is a bijective map that takes advantage of duality between differential forms of degree k and $n-k$ in an n dimensional manifold. For a k-form ω we define $\star\omega$ as the unique $n-k$ form such that

$$\omega \wedge \star\omega = dx_1 \wedge dx_2 \cdots \wedge dx_n = \det$$

where det is the volume form defined by our metric. This operator defines an inner product for differential forms defined by

$$\langle \alpha, \beta \rangle = \bigstar(\alpha \wedge \bigstar\beta)$$

which is easily confirmed to be an inner product (linearity by all operators being linear, positive-definiteness by $\alpha \wedge \bigstar\alpha = 1$).

This pairing is bijective, for as we noted earlier the space of k-forms and (n-k)-forms are the same size, $\binom{n}{n-k}$.

With the Hodge star operator we can now differentiate to the left as well through the following commutative diagram:

$$\begin{array}{ccc} \Omega_k & \underset{d_k}{\overset{\delta_{k+1}}{\rightleftarrows}} & \Omega_{k+1} \\ \downarrow{\scriptstyle \star_k} & & \downarrow{\scriptstyle \star_{k+1}} \\ \Omega_{n-k} & \underset{d_{n-k-1}}{\overset{\delta_{n-k}}{\rightleftarrows}} & \Omega_{n-(k+1)} \end{array}$$

where the bijectivity of the Hodge star allows for us to define the co-differential operator

$$\delta_{k+1} = \star_k^{-1} d_{n-k-1} \star_{k+1} = \star_{n-k} d_{n-k-1} \star_{k+1} .$$

With the codifferential operator we can define more interesting operators Laplace-de Rham operator, which is defined by $\delta d + d\delta$.

A.6 Closed and Exact Forms

Because we now have a few differential operators we can define what closed, exact, co-closed, and co-exact forms are. A k-form α is closed if $d\alpha = 0$ and co-closed if $\delta\alpha = 0$. α is exact if $\alpha = d\beta$ for some $(k-1)$-form β and co-exact if $\alpha = \delta\xi$ for some $(k+1)$-form ξ.

A key fact to note is that every exact form is closed and every co-exact form is co-closed by the fact that $dd = 0$ and $\delta\delta = 0$.

A.6.1 Hodge-Helmholtz Decomposition

A useful tool from this calculus is the Hodge-Helmholtz decomposition, which decomposes any k-form as the sum of three forms: an exact form, a co-exact form, and a harmonic form.

$$\alpha = d\beta + \delta\eta + \gamma$$

This decomposition is unique and guaranteed to exist for any C^2

If we assume that our velocity field is a C^2 1-form u this decomposition gives us the following:

$$u = dp + \delta\omega + \gamma.$$

where p is a 0-form representing pressure, ω is a 2-form representing vorticity, and γ, which is a harmonic form such that $\Delta\gamma = 0$. The flows represented by harmonic forms are the space of Laminar flows and have little relevance to the systems we will be looking at, so we will neglect it.

A.7 Integrating Differential Forms

Before we can uncover how to integrate we need to discuss what we are integrating over. Rather than always integrating over the whole manifold, we will integrate over quantities called *chains*. It's through the discretization of chains into chain complexes that we obtain our numerical methods.

A.7.1 Cube Complexes

Before we define what chains are, lets first talk about cubes and cube complexes. Though the usual discussion on this topic utilizes simplices and simplicial complexes, we choose to use cube complexes because we use cube meshes in our discretization, not simplicial meshes. In this context a k-cube on a manifold M is a chart from $[0,1] \to M$, which we will usually identify with its image on M. When $M = \mathbb{R}^3$ a point is a 0-cube, a line is a 1-cube, a square is a 2-cube, etc.

A k-cube complex is a collection of cubes $\mathcal{S}$ such that the following two statements are true:

- For any two cubes $\alpha, \beta \in \mathcal{S}$, $\alpha \cap \beta$ is a cube in $\mathcal{S}$ of dimension lower than those of α and β.
- For any cube $\alpha \in \mathcal{S}$, $\alpha \subset \beta$ for some k-cube $\beta \in \mathcal{S}$.

The discretization I chose to use was a staggered grid as initially described by [**?**]. The essense of this grid format is that rather than store values on only the vertices of a grid or on the centers, we store quantities on the vertices, cells, and edges. This storage of a n dimensional staggered grid is generated by a highly regular cube complex spanning one larger cube $[0,1]^n$. If we set the number of ticks per axis to be N_i, one i per axis, we get that each n-cube is a translation of $\times[0, \Delta x_i]$, the product of n intervals of width $\Delta x_i = \frac{1}{N_i}$. We will only talk about uniform cubes where all $N_i = N$ for some N. That makes each top-level cube a translation of $[0, \frac{1}{N}]^n$.

A.7.2 Chains

A k-chain is a finite sum of k-cubes with integral coefficients. That means that if we have a couple of k-cubes $c_1, c_2, \cdots, c_l \in \mathcal{S}$ we can generate a k-chain by attaching coefficients $a_1, a_2, \cdots, a_l \in \mathbb{Z}$ to create a k-chain

$$\sigma_k = a_1 c_1 + a_2 c_2 + \cdots + a_l c_l.$$

We will identify individual k-cubes c with the unit k-chain $1c$.

The boundary of each k-cube to its $2k$ (k-1)-cubes on its boundary, so we define the boundary operator ∂ to be the chain generated by the following:

$$\partial c_k = c_{k-1}^1 + c_{k-1}^2 + \cdots + c_{k-1}^{2k}$$

where c_k is a k-cube and the c_{k-1}^i are (k-1)-cubes.

so from any k-chain we can generate a (k-1)-chain through

$$\sigma_{k-1} = \partial\sigma_k = a_1\partial c_1 + a_2\partial c_2 + \cdots + a_l\partial c_l.$$

This boundary operator defines yet another graded algebra, the chain complex, that goes the opposite way of the deRham complex.

$$\mathcal{C}^0 \cdots \leftarrow_{\partial_{n-1}} \mathcal{C}^{n-1} \leftarrow_{\partial_{n-2}} \mathcal{C}^{n-2} \leftarrow_{\partial_n} \mathcal{C}^n$$

where $\mathcal{C}^l \subset \mathcal{S}$ is the set of l-cubes in a k-cube complex.

A.7.2.1 Laplace-de Rham Operator

In the following discussion we will depend heavily on the Laplace-de Rham operator, which is a generalization of standard Laplacian operator to work for differential forms. We'll refer to the Laplace-de Rham as the Laplacian where the situation is unambiguous. It is defined as

$$\mathbb{L}u = (d\delta + \delta d)u$$

where u is a k-form. Because for 0-forms ω and n-forms γ we have

$$\begin{aligned} \delta\omega &= 0 \\ d\gamma &= 0 \end{aligned}$$

In our case we only need the Laplacian of a 0-form, so $\delta u = 0$. This allows us to simplify the Laplacian operators we'll use as

$$\begin{aligned} \mathcal{L}\omega &= \delta d u \\ \mathcal{L}\gamma &= d\delta u \end{aligned}$$

It's fairly clear that $\mathcal{L}$ is self adjoint according to the Riemannian metric γ by the relationship between d and δ and that the Hodge star is self adjoint:

$$\begin{aligned}\langle (d\delta + \delta d)u, v\rangle_\gamma &= \langle d\delta u, v\rangle_\gamma + \langle \delta du, v\rangle_\gamma \\ &= \langle \delta u, \delta v\rangle_\gamma + \langle du, dv\rangle_\gamma \\ &= \langle u, d\delta v\rangle_\gamma + \langle u, \delta dv\rangle_\gamma \\ &= \langle u, (d\delta + \delta d)v\rangle_\gamma\end{aligned}$$

We can see the special n-form and 0-form cases by simply following the left or right sides of the addition above.

In numerical methods solving the Poisson problem ($\Delta u = p$) quite a few methods depend on having a matrix that is self adjoint with respect to the standard metric. In the special cases, however, the Laplace-de Rham matrix is clearly not diagonal with respect to the standard metric:

$$(d \star d^T \star)^T = (d\delta)^T = \delta^T d = \star d \star d^T$$

We therefore call the above the strong Laplace-de Rham operator (or strong Laplacian) and call the following diagonalization of the operator the weak Laplace-de Rham operator (or weak Laplacian):

$$\mathrm{L} = d^T \star d$$

which is clearly self adjoint by the self adjoint-ness of the Hodge star.

A.7.3 Integration

Now that we have a defined differential forms and chains we can finally discuss how one integrates on cube complexes. Integration is done through a pairing between k-chains and differential forms, most easily explained through the constitutive k-cubtes. By abusing notation slightly and referring to k-cubes initially as sets and then as maps from $[0,1]^k \to M$ we obtain that

$$\langle \omega, \sigma\rangle = \int_\sigma \omega = \int_{[0,1]^k} \omega(\sigma(x)) J(x) dx$$

where $\sigma \in \mathcal{C}^k, \omega \in \Omega^k$, and $J(x) = \det(D\sigma(x))$ is the Jacobian of σ evaluated at x.

Under this pairing Stokes theorem becomes the statement that the exterior derivative and boundary operators are each others adjoints:

$$\int_\sigma d\omega = \langle d\omega, \sigma\rangle = \langle \omega, \partial\sigma\rangle = \int_{\partial\sigma} \omega.$$

A.8 Discrete Calculus on Differential Forms

In the previous section we discussed differential forms, but haven't answered why this is relevant to our problem of numerical simulation. The primary reason for all of this is that we really only have two operators to compute in the above system, so by defining discrete analogues to those two operators on

a discrete space should be sufficient. The main question on how to define those two operators is the same as in any problem: where do we store the discrete samples to approximate our space. We choose to follow the practice of DEC (Discrete Exterior Calculus) [HNC08] which is to store objects where they integrally make sense. By that we mean that a k-form is integrated over a k-dimensional manifold so we store k-forms by their evaluated values on k-dimensional manifolds. For instance, if we were using a simplicial complex (say a triangular mesh or a tetrahedreal mesh) 0-forms are stored on points and 2-forms are stored on trianges.

A.8.1 Exterior Derivative and Boundary Operators

This discretization makes generating the exterior derivative rather simple: assume we have k-polyhedron M^k and a differential form σ stored as integrated quantities on the faces M^k, M_i^{k-1} the integrated of $d\sigma$ over M^k. That is, we have some integrated values of σ for each face $\mathcal{M}^{k-1}$ discrete values stored on the faces $\hat{\sigma}_i$ defined by

$$\hat{\sigma}_i = \int_{\mathcal{M}_i^{k-1}} \sigma$$

and by applying Stoke's theorem we have

$$\begin{aligned} \widehat{d\sigma} &= \int_{\mathcal{M}^k} d\sigma \\ &= \sum_i \mathrm{sgn}_i \int_{\mathcal{M}_i^{k-1}} \sigma \\ &= \sum_i \mathrm{sgn}_i \hat{\sigma}_i. \end{aligned}$$

where sgn_i denotes the sign of face M_i^{k-1} with respect to $\mathcal{M}^k$: $+1$ if they share orientation and -1 if not.

This leads to a simple definition of the exterior derivative of a complex of polyhedra: the discrete exterior derivative of every polyhedron is the signed sum of its boundary elements depending on orientation. This is the transpose of the signed boundary operator, which is a natural result from the adjointness of the exterior derivative and the boundary, and so this operator is generally very easy to compute.

The other main component to consider is how to define the Hodge Star in the discrete case.

A.8.2 Hodge Star

In our case we are using a cube complex which means that we are storing quantities on k-cubes. This turns out to be identical to what is commonly used as staggered grid methods, which we discuss in the main body of this thesis.

A.9 Pressure Projection

TODO: Maybe i'll move pressure projection differential forms stuff to here?

Appendix B

Discrete Differential Forms on Cubical Complexes

Appendix A provides the preliminaries of doing exterior calculus on cubical complexes, and here we will discuss the explicit construction of those operators.

We will only consider a particular family of cubical complexes: those that only have vertices from $\mathbb{Z}^3$ and are axis-aligned. This simplification is what will allow us to tie our work with standard staggered-grid methods. Because of our use of a regular cubical grid, there are several convenient simplifications that can be made in implementation. Some of these will be placed in this section while others will be placed in Appendix B.

B.1 Level-set based Hodge Star

Let us begin with the derivations of the basic exerior calculus operators on cubical complexes, which will be used in our discussion of a levelset hodge-star: the Boundary operator and the Hodge star.

Because of the structure of our chosen family of cubical complexes there is a very convenient indexing scheme that we will use for the rest of this article: If in $\mathbb{R}^n$ we let a $(i, j, k) \in \mathbb{Z}^3$ denote the index of the unit n-cube that is centered at that location, we can uniquely index every k-cube by the location of center. The set of centers is isomorphic to $(\mathbb{Z}/2)^n$. In fact the centers of all of the k-cubes can be represented by the subset of $(\mathbb{Z}/2)^n$ such that k of the numbers in the n-tuple are integral (i.e $(2, 3, 4.5)$ represents a 2-form and $(0, 0, 0)$ represents a 3-form). From herein we will denote a n-cube by its index.

B.1.1 Connection to Staggered Grid Methods

In a staggered grid configuration one usually maintains a couple of grid types: one for vertices, one for cells, and n grids for fluxes along each coordinate axis.

The cell grid is usually used to represent pressure, which is a volumetric quantity and therefore representable as a n-form. The flux grids are used to represent one coordinate of the velocity at a time and are exactly where our $n-1$ forms are placed — and a common interpretation of $n-1$ forms is flux. The forms of lower dimension are usually not used.

B.1.2 Boundary Operator

Because of the regularity of our cubical complexes (the vertices form a lattice), Each n-cube is surrounded by $2*n$ edges, two per coordinate axis. Per axis, one of the boundary $n-1$ cubes has a lower index in the grid and one has a higher index in that chosen axis. We choose to give the lower index a different sign and the higher index the same sign. For a cube (i, j, k) this makes the relevant entries in the boundary map the following:

$$\begin{aligned}
\partial_{(i,j,k),(i-\frac{1}{2},j,k)} &= -1 \\
\partial_{(i,j,k),(i+\frac{1}{2},j,k)} &= 1 \\
\partial_{(i,j,k),(i,j-\frac{1}{2},k)} &= -1 \\
\partial_{(i,j,k),(i,j+\frac{1}{2},k)} &= 1 \\
\partial_{(i,j,k),(i,j,k-\frac{1}{2})} &= -1 \\
\partial_{(i,j,k),(i,j,k+\frac{1}{2})} &= 1.
\end{aligned}$$

In implementation this is usually performed on a 3-dimensional array of cubes forming a filled rectangular prism. If we set the height, width, and depth of this prism to by M, N, O respectively we obtain a boundary operator matrix with dimensions NMO by $(N+1)MO + N(M+1)O + NM(O+1)$.

B.1.3 Hodge Star

First let us go over how the Hodge Star on uniform grids is the same as standard finite difference operators.

B.1.3.1 Hodge Star on Uniform Grids

On a staggered grid with uniform spacing the diagonal Hodge Star turns out to be scalar. This begins with remembering that the diagonal Hodge Star is defined as the ratio of the volumes of the cube and then noting that the volume of a k-cube is Δx^k, so the diagonal Hodge Star on is

$$\bigstar^k = \frac{\Delta x^{n-k}}{\Delta x^k}.$$

$$\begin{aligned}
\bigstar^n &= \frac{1}{\Delta x^n} \\
\bigstar^{n-1} &= \frac{\Delta x^1}{\Delta x^{n-1}}.
\end{aligned}$$

B.1.3.2 Differential and Codifferential Operator

Now that we have the boundary and the differential operator are adjoint, we see that the discrete representation of the differential operator is

$$d = \partial^*.$$

Then if we remind ourselves that the codifferential operator is defined by $\delta^n = (\bigstar^{n-1})^{-1}(d^n)^*\bigstar^n$ we quickly see that

$$\delta^n = (\bigstar^{n-1})^{-1}(d^n)^*\bigstar^n = \frac{\Delta x^{n-1}}{\Delta x}(d^n)^*\frac{1}{\Delta x^n} = \frac{\Delta x^{n-1}}{\Delta x}\frac{1}{\Delta x^n}(d^n)^* = \frac{1}{\Delta x^2}(d^n)^*$$

so the codifferential operator is simply the boundary operator scaled by $\frac{1}{\Delta x^2}$.

B.1.3.3 Constructing the Laplace-deRham

For n-forms the Laplace-deRham operator is the standard finite difference Laplacian, with a 5-point stencil in 2 dimensions and 7-point stencil in 3 dimensions comprising of $-\frac{1}{\Delta x^2}$ off diagonal bands and positive entries to main diagonal to make each row sum zero. To see this consider the following:

First the explicit representation of Laplace-deRham is

$$d\delta = \frac{1}{\Delta x^2}dd^T = \frac{1}{\Delta x^2}\partial^T\partial.$$

To compute the value for the row representing $\alpha = (i, j, k)$ and column representing $\hat{\alpha} = (\hat{i}, \hat{j}, \hat{k})$ recall from linear algebra that the entry for $\alpha, \hat{\alpha}$ is $\partial_{\alpha,:} \cdot \partial_{\hat{\alpha},:}$ where $\partial_{a,:}$ is vector of the a column in ∂. With that information we can directly compute the entries of the matrix from three cases:

- $\alpha = \hat{\alpha}$

  Since all entries line up and all of the entries are ± 1 we get $2 * n$ nonzero entries: one for each face on the boundary of the cube. Therefore $(d\delta)_{\alpha,\hat{\alpha}} = \frac{1}{\Delta x^2}2n$

- α shares a face with $\hat{\alpha}$

  Here the only shared nonzero entries are those representing the shared face. This face, by construction, is the upper face of one and the lower face of another, so the signs of the only nonzero entries are different. Therefore $(d\delta)_{\alpha,\hat{\alpha}} = -\frac{1}{\Delta x^2}$

- α shares no faces with $\hat{\alpha}$.

  Since no nonzero entries coincide the dot product must result in 0.

Appendix C

Numerical Optimization

In order to implement the methods described above it was necessary to implement and utilize a Quadratic Programming solver in two separate locations, with one of the systems actually being a Linear Complementary Problem. Because of the structure of systems being solved, it seemed prudent to use one particular solver called Modified Proportions with Reduced Gradient Projections (MPRGP) and there was need to modify the solver to make it usable in our systems. Some of these modifications were utilized in [NGCL09] but not detailed in the paper, but we will discuss them fully in the technical details section. Therefore, we thought it prudent to provide some introduction to LCP and QP problems and solvers.

First we will discuss some basic notations of constrained optimization. We will then overview the formulations of LCP/QP problems. Finally, we will discuss the duality between LCP and QP problems, which allows for us to use one solver for two types of systems.

C.0.4 Constrained Optimization

Constrained optimization is the task of minimizing some function $f : \mathbb{R}^n \mapsto \mathbb{R}$ under equality and inequality constraints. Let us notate these constraints all as c_i where $i \in \mathcal{E}$ are equality constraints and $i \in \mathcal{I}$ are inequality constraints.

The feasible set, the set of legitimate values for x in this constrained space, is defined as

$$\Omega = \{x | c_i(x) = 0, i \in \mathcal{E}, c_i(x) \geqslant 0, i \in \mathcal{I}\}$$

The boundary of Ω is quite important so there's a fair bit of terminology around it. The active set $\mathcal{A}(x)$ is the set of constraints for which $c_i(x) = 0$ and $\mathcal{F}(x)$, the inactive set, denotes the complement of $\mathcal{A}(x)$. The set of constraints that are active becomes a fundamental tool in the algorithms to come.

The Karush-Kuhn-Tucker (KKT) conditions are first order necessary conditions for a local minimum for an optimization problem with equality and inequality constraints and can be seen as a generalization

of Lagrange multipliers to support inequality constraints. They can be seen as the system

$$\begin{aligned} \nabla_x \mathcal{L}(x,\lambda) &= 0 \\ c_i(x) &= 0,\ \forall i \in \mathcal{E} \\ c_i(x) &\geqslant 0,\ \forall i \in \mathcal{I} \\ \lambda_i &\geqslant 0,\ \forall i \in \mathcal{I} \\ \lambda_i c_i(x) &\geqslant 0,\ \forall i \in \mathcal{E} \cup \mathcal{I} \end{aligned}$$

where ∇_x is the gradient with respect to x and

$$\mathcal{L}(x,\lambda) = f(x) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(x).$$

C.0.5 Linear Complementary Problems

In my exploration of LCP solvers two distinct formulations of the problem definition that were used to develop algorithms. The standard formulation of linear complimentary problems is given by

$$\begin{aligned} w &= Mz + q && \text{(C.1)} \\ w &\geqslant 0 && \text{(C.2)} \\ z &\geqslant 0 && \text{(C.3)} \\ w^T z &= 0 && \text{(C.4)} \end{aligned}$$

where M usually assumed to be symmetric positive definite. The SPD assumption exists because solutions are only known if it is SPD. The reason for this assumption can be understood through the connection between LCP and quadratic programming. Any LCP instance can be seen a the solution to the Karush-Kunn-Tucker (KKT) constraints of the quadratic problem instance

$$\min_u \phi(z) = \frac{1}{2} z^T M z + q^T z \tag{C.5}$$

given inequality constraints that $z_i \geqslant 0$ for all i. If we allow for equality constraints we obtain the mixed LCP (mLCP) formulation of

$$\begin{aligned} Au + Cv + a &= 0 && \text{(C.6)} \\ C^T u + Bv + b &= w && \text{(C.7)} \\ v^T w &\geqslant 0 && \text{(C.8)} \\ v, w &\geqslant 0 && \text{(C.9)} \\ & && \text{(C.10)} \end{aligned}$$

which solves a slightly different quadratic system:

$$\min_z \phi(z) = \frac{1}{2} z^T Q z + r^T z \tag{C.11}$$

where $v \geqslant 0$ and

$$Q = \begin{bmatrix} A & C \\ C^T & B \end{bmatrix} z = \begin{bmatrix} u \\ v \end{bmatrix}, \quad r = \begin{bmatrix} a \\ b \end{bmatrix}.$$

The equivalence of the two formulations is rather easy to formulate. Given an LCP (M, q) we can create an equivalent mLCP by setting (A, B, C, a, b) to be $(0, M, 0, 0, q)$.

An mLCP can be reduced to an LCP by rearranging terms in the mLCP formulation by solving for u and substituting,which provides us with:

$$M = B - C^T A^{-1} C \tag{C.12}$$
$$q = = b - C^T A^{-1} a \tag{C.13}$$

.

C.0.6 LCP and Karush-Kuhn-Tucker Conditions

Though I've mentioned that LCPs can be written as the solutions to a quadratic programming instance, it can also be shown that the KKT first order conditions for a quadratic programming instance can easily be described in terms of an LCP. The natural instance of a quadratic programming problem is a quadratic functional bounded by linear inequality constraints like this:

$$\min_x \quad \frac{1}{2} x^T G x + x^T c \tag{C.14}$$
$$Ax \geqslant b \tag{C.15}$$
$$\tag{C.16}$$

Under the standard LCP formulation the only bound on x is positive and other constraints are not obviously represented. By applying the KKT conditions on this system, we see that a solution to the above quadratic programming instance must be a solution to the LCP given by the system below where a slack variable λ must be introduced:

$$\begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix}.$$

Given that if the above system has full rank and G is positive symmetric definite the solution is unique and so solving the LCP is equivalent to solving the quadratic problem. Therefore we see that LCP is equivalent to QP.

C.0.7 Complimentary and Minimization Duality

Maintaining the complimentary condition in an iterative process is quite difficult. When the constraint variables are strictly positive instead of directly forcing the complimentary condition in iterative algorithms, it's simpler to treat the constraint as a minimization problem. That is, two vectors $u, v \geqslant 0$ are complimentary to each other if and only if for any index i, $\min(u_i, v_i) = 0$ so we can reformulate our complimentary equality constraint as a minimization constraint on the min of the two vectors coefficient-wise. This will be used to determine convergence tests in the iterative algorithms that follow.

Bibliography

[BB09] Tyson Brochu and Robert Bridson. Robust topological operations for dynamic explicit surfaces. *SIAM Journal on Scientific Computing*, 31(4):2472–2493, 2009.

[BBB07] Christopher Batty, Florence Bertails, and Robert Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Transactions on Graphics (TOG)*, 26(3):100, 2007.

[BBS00] SG Bardenhagen, JU Brackbill, and Deborah Sulsky. The material-point method for granular materials. *Computer methods in applied mechanics and engineering*, 187(3):529–541, 2000.

[Bea72] Jacob Bear. *Dynamics of fluids in porous media*. Courier Dover Publications, 1972.

[BMF07] Robert Bridson and Matthias Müller-Fischer. *Fluid simulation: SIGGRAPH 2007 course notes Video files associated with this course are available from the citation page*. ACM, 2007.

[Bri07] Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH*, volume 2007, page 5, 2007.

[BT07] Markus Becker and Matthias Teschner. Weakly compressible sph for free surface flows. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 209–217. Eurographics Association, 2007.

[BYM05] Nathan Bell, Yizhou Yu, and Peter J Mucha. Particle-based simulation of granular materials. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 77–86. ACM, 2005.

[Cho67] Alexandre Joel Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of computational physics*, 2(1):12–26, 1967.

[DG96] Mathieu Desbrun and Marie-Paule Gascuel. *Smoothed particles: A new paradigm for animating highly deformable bodies*. Springer, 1996.

[DS05] Zdenek Dostál and Joachim Schoberl. Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination. *Computational Optimization and Applications*, 30(1):23–43, 2005.

[EFFM02] Douglas Enright, Ronald Fedkiw, Joel Ferziger, and Ian Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83–116, 2002.

[EMF02] Douglas Enright, Stephen Marschner, and Ronald Fedkiw. Animation and rendering of complex water surfaces. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 736–744. ACM, 2002.

[FSJ01] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM, 2001.

[GBO04] Tolga G Goktekin, Adam W Bargteil, and James F O'Brien. A method for animating viscoelastic fluids. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 463–468. ACM, 2004.

[GM77] Robert A Gingold and Joseph J Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389, 1977.

[Har25] F Hardy. Cohesion in colloidal soils. *J. Agric. Sci*, 15:419, 1925.

[HNC08] Anil N Hirani, Kalyana B Nakshatrala, and Jehanzeb H Chaudhry. Numerical method for darcy flow derived using discrete exterior calculus. *arXiv preprint arXiv:0810.3434*, 2008.

[HW+65] Francis H Harlow, J Eddie Welch, et al. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of fluids*, 8(12):2182, 1965.

[ICS+13] Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. Implicit incompressible sph. 2013.

[IWT13] Markus Ihmsen, Arthur Wahl, and Matthias Teschner. A lagrangian framework for simulating granular material with high detail. *Computers & Graphics*, 37(7):800–808, 2013.

[KP06] Danny M Kaufman and Dinesh K Pai. Randomized quadratic programming with applications to rigid body contact. 2006.

[LAD08] Toon Lenaerts, Bart Adams, and Philip Dutré. Porous flow in particle-based fluid simulations. In *ACM Transactions on Graphics (TOG)*, volume 27, page 49. ACM, 2008.

[LD09] Toon Lenaerts and Philip Dutré. Mixing fluids and granular materials. In *Computer Graphics Forum*, volume 28, pages 213–218. Wiley Online Library, 2009.

[Llo05] John E Lloyd. Fast implementation of lemke's algorithm for rigid body contact simulation. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 4538–4543. IEEE, 2005.

[NGCL09] Rahul Narain, Abhinav Golas, Sean Curtis, and Ming C Lin. Aggregate dynamics for dense crowd simulation. In *ACM Transactions on Graphics (TOG)*, volume 28, page 122. ACM, 2009.

[NGL10] Rahul Narain, Abhinav Golas, and Ming C Lin. Free-flowing granular materials with two-way solid coupling. In *ACM Transactions on Graphics (TOG)*, volume 29, page 173. ACM, 2010.

[RSKN08] Witawat Rungjiratananon, Zoltan Szego, Yoshihiro Kanamori, and Tomoyuki Nishita. Real-time animation of sand-water interaction. In *Computer Graphics Forum*, volume 27, pages 1887–1893. Wiley Online Library, 2008.

[RWT11] Karthik Raveendran, Chris Wojtan, and Greg Turk. Hybrid smoothed particle hydrodynamics. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 33–42. ACM, 2011.

[SKV+12] Breannan Smith, Danny M Kaufman, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. Reflections on simultaneous impact. *ACM Transactions on Graphics (TOG)*, 31(4):106, 2012.

[SOH99] Robert W Sumner, James F O'Brien, and Jessica K Hodgins. Animating sand, mud, and snow. In *Computer Graphics Forum*, volume 18, pages 17–26. Wiley Online Library, 1999.

[SP09] Barbara Solenthaler and Renato Pajarola. Predictive-corrective incompressible sph. In *ACM transactions on graphics (TOG)*, volume 28, page 40. ACM, 2009.

[SSC+13] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)*, 32(4):102, 2013.

[Sta99] Jos Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 121–128. ACM Press/Addison-Wesley Publishing Co., 1999.

[WMFB11] Chris Wojtan, Matthias Müller-Fischer, and Tyson Brochu. Liquid simulation with mesh-based surface tracking. In *ACM SIGGRAPH 2011 Courses*, page 8. ACM, 2011.

[YHK08] Ren Yasuda, Takahiro Harada, and Yoichiro Kawaguchi. Real-time simulation of granular materials using graphics hardware. In *Computer Graphics, Imaging and Visualisation, 2008. CGIV'08. Fifth International Conference on*, pages 28–31. IEEE, 2008.

[YT13] Jihun Yu and Greg Turk. Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Transactions on Graphics (TOG)*, 32(1):5, 2013.

[ZB05] Yongning Zhu and Robert Bridson. Animating sand as a fluid. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 965–972. ACM, 2005.

[ZOF01] Hong-Kai Zhao, Stanley Osher, and Ronald Fedkiw. Fast surface reconstruction using the level set method. In *Variational and Level Set Methods in Computer Vision, 2001. Proceedings. IEEE Workshop on*, pages 194–201. IEEE, 2001.

[ZY10] Bo Zhu and Xubo Yang. Animating sand as a surface flow. *Eurographics 2010, Short Papers*, 2010.