Drucker-Prager Elastoplasticity for Sand Animation

Gergely Klár Theodore Gast Andre Pradhana Chuyuan Fu Craig Schroeder Chenfanfu Jiang Joseph Teran
University of California, Los Angeles

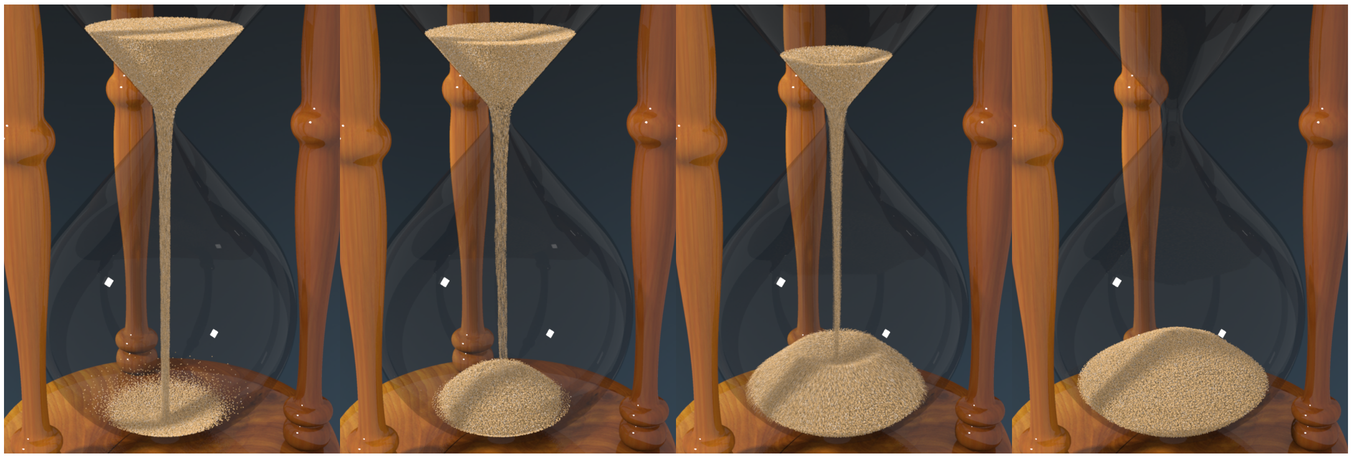

Figure 1: *Sand falls through the narrow neck of an hourglass, accumulating at the bottom.*

Abstract

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

Keywords: MPM, APIC, elastoplasticity, sand, granular

Concepts: **•Computing methodologies → Physical simulation;** *Continuous simulation;*

1 Introduction

Sand dynamics are ubiquitous in every day environments. Its characteristic flowing and piling motions must be recreated with a high level of accuracy when animating scenes like beaches or playgrounds. Sand and many other similar everyday materials like salt, powder, rubble, etc are granular materials composed of many discrete macroscopic grains colliding and sliding against one another. These materials exhibit complex behaviors with aspects comparable to both fluids (e.g. they can assume the shape of a container) and solids (they can support weight and form stable piles) [Jaeger et al. 1996; Bardenhagen et al. 2000].

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

We build on the work of Mast et al. [2013; 2014] and develop an implicit version of their Drucker-Prager-based elastoplasticity model for granular materials. The Drucker-Prager conception of elastoplasticity is often used in the mechanical engineering literature for granular materials [Drucker and Prager 1952], and we show that it can be adopted to animation applications with relatively simple implementation and efficient runtimes. This is useful because the models are well developed and the literature can be consulted to reduce the difficulty of parameter tuning. We use the Material Point Method (MPM) [Sulsky et al. 1994] to discretize the model since it provides a natural and efficient way of treating contact, topological change and history dependent behavior. Furthermore, we show that this can be done with little more effort than was used for simulating snow dynamics in the MPM approach of Stomakhin et al. [2013]. Lastly, we replace the particle/grid transfers used by Mast et al. with APIC transfers [Jiang 2015; Jiang et al. 2015] and show that

SIGGRAPH '16 Technical Paper,, July 24 - 28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925906

this allows for more stable behavior, particularly with simulations that have higher numbers of particle per cell.

2 Related Work

Continuum approaches have been used in a number of graphics methods for granular materials. Zhu and Bridson [2005] animate sand as a continuum with a modified Particle-In-Cell fluid solver. Narain et al. [2010] improve on the method of Zhu and Bridson by removing cohesion artifacts associated with incompressibility. Both of these works led to a number of generalizations and improvements. Nkulikiyimfura et al. [2012] develop a GPU version of the Zhu and Bridson approach. Laenerts and Dutre [2009] use an SPH version to couple water with porous granular materials. Alduán and Otaduy [2011] generalize the unilateral incompressibility developed by Narain et al. to SPH. Imhsen et al. [2013] show how to improve the convergence of the method of Alduán and Otaduy [2011] and also detail refinement of base simulations to upscale to millions of grains. Chang et al. [2012] use a modified Hooke's law to handle friction between grains.

Many methods are developed by modeling interactions between individual grains or particle idealizations of grains, rather than from a continuum. Miller and Pearce [1989] simulate interactions between particles to model sand, solid and viscous behaviors. Luciani et al. [1995] use a similar approach. Bell et al. [2005] got very impressive results by simulating many sand grains as spherical rigid bodies with friction. Milenkovic [1996] also simulated individual grains to solve for piles of rigid materials via energy minimization/optimization. Mazhar et al. [2015] use Nestov's method to simulate millions of individual grains. Yasuda et al. [2008] use the GPU to get real-time results with rigid grains. Alduan et al. [2009] use an adaptive resolution version of the method by Bell et al. [2005] to improve performance. Macklin et al. [2014] show that the extremely efficient position based dynamics methods can be applied by casting granular interactions as hard constraints in.

As a method that combines aspects of particle-based and grid-based methods, MPM has proven to be a useful discretization choice for granular materials, such as snow [Stomakhin et al. 2013] and sand [Jiang et al. 2015]. It has also been used for more general elastoplastic flows for computer graphics [Yue et al. 2015; Stomakhin et al. 2014; Ram et al. 2015; Jiang 2015; Jiang et al. 2015] and engineering applications [Mast 2013; Mast et al. 2014]. Brackbill et al. [Bardenhagen et al. 2000; Cummins and Brackbill 2002] simulate individual grains but use MPM to resolve collisions and friction.

When extreme computational efficiency is required, simplified approaches like height fields [Sumner et al. 1999; Onoue and Nishita 2003; Li and Moshell 1993; Chen and Wong 2013; Chanclou et al. 1996] and cellular automata [Pla-Castells et al. 2006] have been used to provide real-time interaction.

3 Background

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

$$\frac{D\rho}{Dt} + \rho\nabla \cdot \mathbf{v} = 0 \tag{1}$$

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

$$\rho\frac{D\mathbf{v}}{Dt} = \nabla \cdot \boldsymbol{\sigma} + \rho\mathbf{g}. \tag{2}$$

Here, we have used $\frac{D\phi}{Dt} = \frac{\partial\phi}{\partial t} + \mathbf{v} \cdot \nabla\phi$ to denote the material derivative of an arbitrary function $\phi(x, y, z)$. The text of Gonzalez and Stuart [2008] provides useful background for these equations.

Deformation gradient. The deformation gradient represents how deformed a material is locally. For example, let $\mathbf{x}_1^0$ and $\mathbf{x}_2^0$ be two nearby points embedded in the material (see Figure 4) at the beginning of the simulation, and let $\mathbf{x}_1$ and $\mathbf{x}_2$ be the same two points in the current configuration. Then $(\mathbf{x}_2 - \mathbf{x}_1) = \mathbf{F}(\mathbf{x}_2^0 - \mathbf{x}_1^0)$. The deformation gradient $\mathbf{F}$ evolves according to

$$\frac{D\mathbf{F}}{Dt} = (\nabla\mathbf{v})\mathbf{F}. \tag{3}$$

Elastic and plastic deformation gradient. We represent plasticity by factoring deformation gradient into elastic and plastic parts as $\mathbf{F} = \mathbf{F}^E\mathbf{F}^P$. The deformation gradient is a measure of how a material has locally rotated and deformed due to its motion. By factoring the deformation gradient in this way, we divide this deformation history into two pieces. The plastic part, $\mathbf{F}^P$, represents the portion of the material's history that has been forgotten. If a metal rod is bent into a coiled spring, the rod *forgets* that it used to be straight; the coiled spring behaves as though it was always coiled (see Figure 6). The twisting and bending involved in this operation is stored in $\mathbf{F}^P$. If the spring is compressed slightly, the spring will feel strain (deformation). This is elastic deformation, which is stored in $\mathbf{F}^E$. The spring remembers this deformation. In response, the material exerts stress to try to restore itself to its coiled shape. In this way, we see that only $\mathbf{F}^E$ should be used to compute stress. The full history of the metal rod consists of being bent into a spring shape ($\mathbf{F}^P$) and then being compressed ($\mathbf{F}^E$).

Constitutive model. A constitutive model relating the state to the stress is needed. The text of Bonet and Wood [2008] provides useful background for elastoplastic constitutive modeling. Following Mast el al. [2013; 2014], we use Drucker-Prager elastoplasticity [Drucker and Prager 1952]. The elastic part of this relation is expressed through the deformation gradient $\mathbf{F}$.

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

$$\boldsymbol{\sigma} = \frac{1}{\det(\mathbf{F})}\frac{\partial\psi}{\partial\mathbf{F}^E}\mathbf{F}^{E^T} \tag{4}$$

where $\psi(\mathbf{F}^E)$ is the elastic energy density designed to penalize non-rigid $\mathbf{F}^E$ (see Section 6.3 for more detailed discussion).

With the Drucker-Prager model, frictional interactions between grains of sand can be expressed in the continuum via a relation between shear and normal stresses. Using a Coulomb friction model, shear stresses resisting sliding motions between grains can only be as large as a constant times the normal stress holding them together. For example, if shear stresses larger than this value are required to maintain a static pile, plastic flow will commence when the limit is reached and the material will move. This constraint defines a feasible region of stresses, the surface of the feasible region is often referred to as the yield surface. The decomposition of the deformation gradient into elastic and plastic components $\mathbf{F} = \mathbf{F}^E\mathbf{F}^P$ can

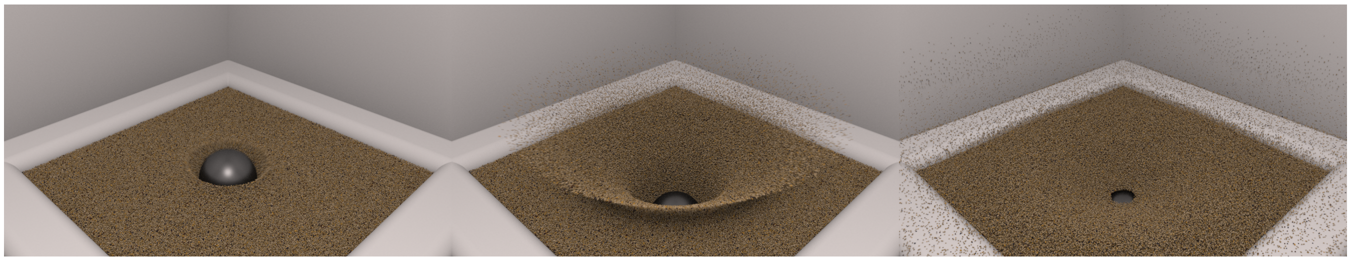

Figure 2: *A solid ball drops into a sandbox, spraying sand in all directions.*

be viewed as a means of projecting the deformation to satisfy the constraint. Notably, the projection must be designed carefully to ensure for increase of entropy as well as volume preserving plastic flow. We discuss the model in more detail in Section 7 as well as in the supplementary technical document [Klár et al. 2016].

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

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

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

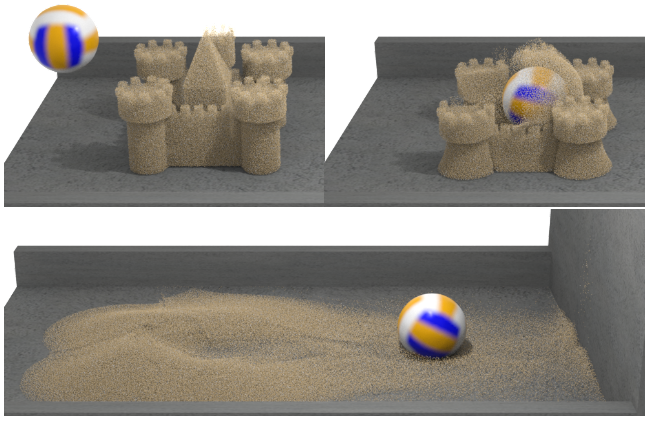

Figure 3: *A sand castle is hit with a deformable ball while falling. The sand and ball are fully coupled in the simulation.*

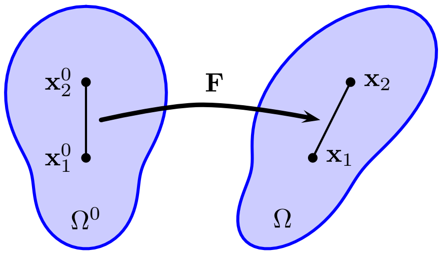

Figure 4: *Relationship between deformation and* **F**.

The Material Point Method stores information on Lagrangian particles, but it computes forces using a fixed Eulerian grid. The use of particles makes mass conservation trivial, and it provides a simple means of moving information around. The use of a fixed grid provides automatic handling of topology changes (merging and separating) and collisions between regions of material. Since MPM uses two distinct representations, information must be transferred between them. These transfers play a very important role in the numerical behavior of a hybrid method. Furthermore, to simplify topology changes, MPM does not store connectivity between particles. This avoids the need for complex remeshing, but deformation must now be tracked in an Eulerian way.

In the next section, we outline our discretization steps.

4 Overview

Before presenting the algorithm in detail, we first provide an overview of the steps that are involved in the algorithm and the role that they play, which is summarized in Figure 7.

1. **Transfer to grid.** Transfer mass and momentum from particles to the grid. Use mass and momentum to compute velocity on the grid. (§5.1)
2. **Apply forces.** Compute elastic forces using a deformation gradient that has been projected into the plastic yield surface and apply the forces to the grid velocities. (§6)
3. **Grid collisions.** Project grid velocities for collisions against scripted bodies and obstacles, ignoring friction (§8). For implicit, this is merged with the force application step (§5.6).
4. **Friction.** Compute and apply friction based on the collisions that were resolved. The velocity before and after this step are retained for use during the transfers. (§8.1)
5. **Transfer to particles.** Transfer velocities from grid to particles, being careful to handle friction in a manner that does not lead to inconsistencies. (§5.3)
6. **Update particles.** Update remaining particle state, including positions and deformation gradient. (§5.4)
7. **Plasticity and hardening.** Project the deformation gradient for plasticity, updating the elastic and plastic parts. Perform hardening, which updates the plastic yield surface. (§5.5)

5 Algorithm

Notation. It is helpful to establish the conventions for notation (see Table 1 for a complete list). Scalars are represented by non-bold Latin or Greek characters (m_p, α_p^n, G_k). Vectors are represented

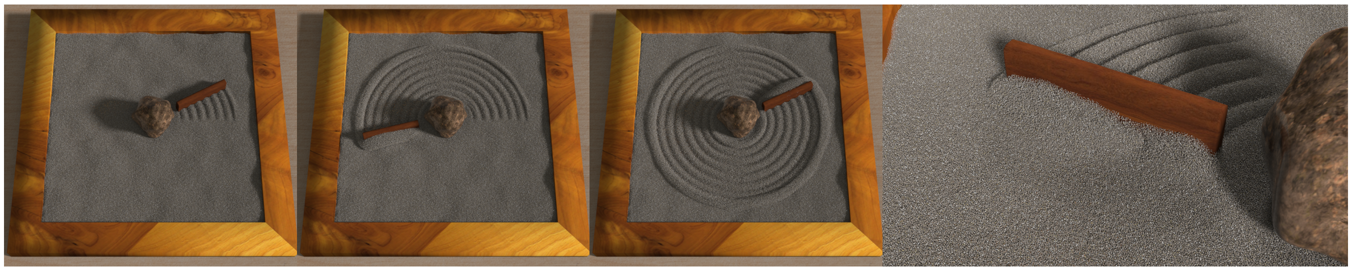

Figure 5: *A rake is dragged around a rock, producing a circular pattern in the sand.*

by bold lowercase Latin characters ($\mathbf{v}_p^n$, $\overline{\mathbf{x}}_i^{n+1}$). Matrices are represented by bold uppercase Latin characters or bold Greek characters ($\mathbf{I}$, $\hat{\mathbf{F}}_p^{P,n+1}$, $\boldsymbol{\sigma}$). Derivatives alter this in the usual way, so that ∇G_{ki} is a vector and $(\nabla \mathbf{v})_p$ is a matrix.

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

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

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

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

Generally, quantities that are stored on particles have indicators of time, and those that lack other adornments are state variables (α_p^n, $\mathbf{F}_p^{E,n}$, $\mathbf{x}_p^n$; not $\mathbf{v}_i^n$, $\hat{\mathbf{F}}_p^{E,n+1}$). There are two exceptions to this. We do not store the deformation gradient itself, so $\mathbf{F}_p^n$ for us is not a state variable. The mass m_p is a state variable, but we omit a time indicator because it never changes. State variables are those that persist from the end of one time step to the beginning of the next.

Particle state. In MPM, the primary representation of state is stored on particles. We maintain mass m_p, position $\mathbf{x}_p^n$, velocity $\mathbf{v}_p^n$, and affine momentum $\mathbf{B}_p^n$, which is related to the velocity spatial derivatives. The extra matrix $\mathbf{B}_p^n$ stored per particle is used for APIC transfers [Jiang et al. 2015]. Up to a constant scale, this quantity approximates the spatial derivative of the grid velocity field at the end of the previous time step.

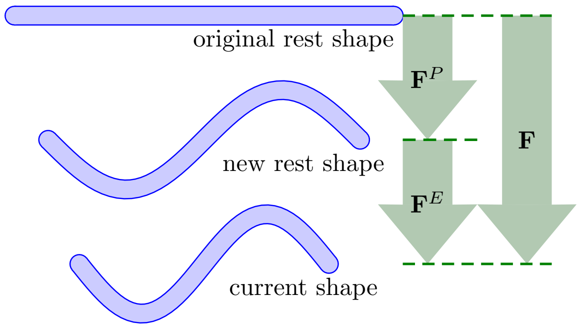

Figure 6: *Relationship between* $\mathbf{F}$, $\mathbf{F}^E$, *and* $\mathbf{F}^P$.

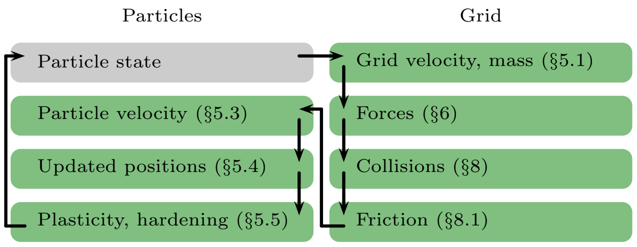

Figure 7: *Overview of MPM stages.*

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

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

Choosing a kernel N leads to trade offs with respect to smoothness, computational efficiency, and the width of the stencil. We prefer tensor product splines for their computational efficiency, as they are relatively inexpensive to compute, differentiate, and store. The multilinear kernel typically employed for FLIP fluid solvers is the simplest of these options, but it is not suitable here. There are two reasons for this (see [Steffen et al. 2008]). The first is that ∇w_{ip}^n would be discontinuous and produce discontinuous forces. The second is that ∇w_{ip}^n may be far from zero when $w_{ip}^n \approx 0$, leading to large forces being applied to grid nodes with tiny weights. Quadratic and cubic b-splines work well, and we choose cubic b-splines for convenience. Our kernel is

$$\hat{N}(x) = \begin{cases} \frac{1}{2}|x|^3 - |x|^2 + \frac{2}{3} & 0 \le |x| < 1 \\ \frac{1}{6}(2-|x|)^3 & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases} \tag{5}$$

$$N(\mathbf{u}) = \hat{N}\left(\frac{u_x}{h}\right)\hat{N}\left(\frac{u_y}{h}\right)\hat{N}\left(\frac{u_z}{h}\right), \tag{6}$$

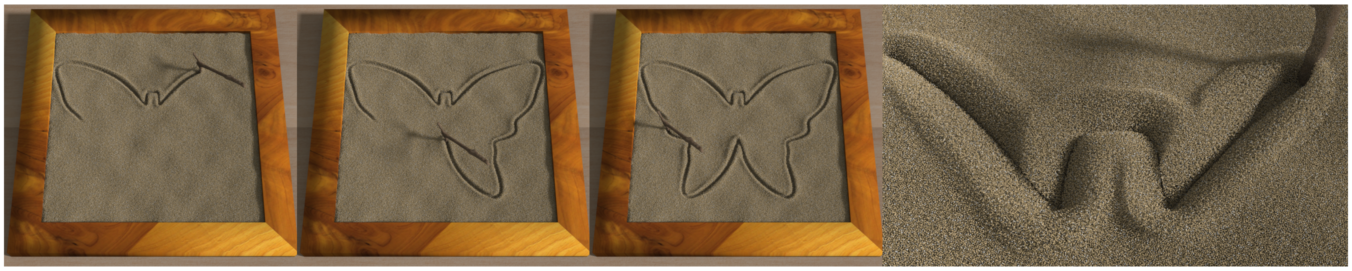
Figure 8: *A stick is dragged through a bed of sand, tracing out a butterfly shape in the sand.*

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

Table 1: *Table of notation used in this paper.* [†]*These quantities are state on particles.*

where h is the grid spacing. Sometimes the quadratic kernel is also useful. We plot the quadratic and cubic kernels in Figure 9. We use the cubic spline for all of our examples.

5.1 Transfer to grid

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

$$m_i^n = \sum_p w_{ip}^n m_p \tag{7}$$

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

The next task is to transfer velocity. We do this using the APIC transfers in [Jiang et al. 2015]. The velocity state on the particle is represented by $\mathbf{v}_p^n$ and $\mathbf{B}_p^n$. The affine momentum $\mathbf{B}_p^n$ is related to the velocity spatial derivatives $\mathbf{C}_p^n$ through $\mathbf{C}_p^n = \mathbf{B}_p^n(\mathbf{D}_p^n)^{-1}$, where $\mathbf{D}_p^n$ is a matrix that behaves as an inertia tensor and is

$$\mathbf{D}_p^n = \sum_i w_{ip}^n(\mathbf{x}_i^n - \mathbf{x}_p^n)(\mathbf{x}_i^n - \mathbf{x}_p^n)^T = \begin{cases} \frac{h^2}{3}\mathbf{I} & \text{cubic} \\ \frac{h^2}{4}\mathbf{I} & \text{quadratic} \end{cases} \tag{8}$$

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

With $\mathbf{C}_p^n$, we can define an affine velocity field $\overline{\mathbf{v}}_p(\mathbf{x})$ for particle p by $\overline{\mathbf{v}}_p(\mathbf{x}) = \mathbf{v}_p + \mathbf{C}_p^n(\mathbf{x} - \mathbf{x}_p^n)$. The momentum contribution from

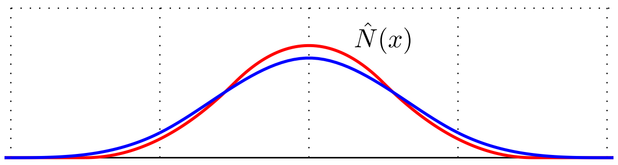

Figure 9: *Cubic (blue) and quadratic (red) splines used for computing interpolation weights.*

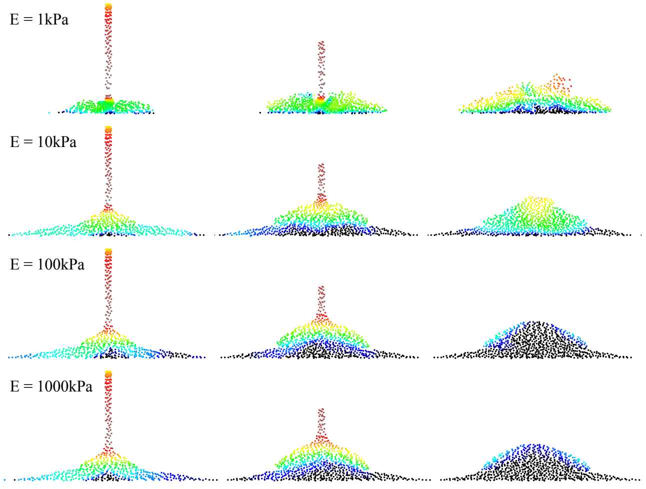

Figure 10: *This simulation shows the effects of Young's modulus on the behavior of a simulation. Sand with a very low Young's modulus tends to be bouncy. The behavior is more like sand as the Young's modulus approaches its physical value.*

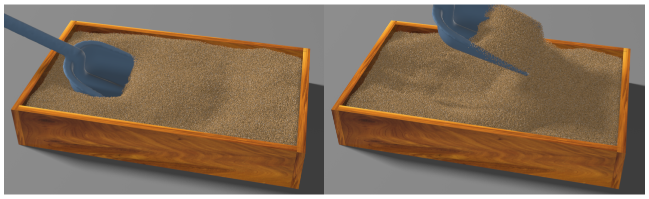

Figure 11: *A shovel digs through sand and pushes it aside.*

particle p to node i is $w_{ip}^n m_p \overline{\mathbf{v}}_p(\mathbf{x}_i^n)$. This leads to the full form of the velocity transfer,

$$\mathbf{v}_i^n = \frac{1}{m_i^n} \sum_p w_{ip}^n m_p (\mathbf{v}_p^n + \mathbf{B}_p^n (\mathbf{D}_p^n)^{-1} (\mathbf{x}_i^n - \mathbf{x}_p^n)) \qquad (9)$$

5.2 Grid update

We next update velocities on the grid. This involves applying forces and processing for collisions with scripted objects. We present two approaches for doing this, explicit and implicit. For most of our examples, explicit is more efficient, since we are running with relatively low stiffness. For stiff examples, implicit becomes advisable. We defer the implicit formulation until (§5.6).

Explicit The simplest approach for handling forces is explicit. In this case, we compute and apply an explicit force (§6)

$$\mathbf{v}_i^\star = \mathbf{v}_i^n + \frac{\Delta t}{m_i^n} \mathbf{f}_i(\langle \mathbf{F}_p^{E,n} \rangle). \qquad (10)$$

After forces are applied, we can process the velocities for collisions $\mathbf{v}_i^\star \rightarrow \overline{\mathbf{v}}_i^{n+1}$ and then apply friction $\overline{\mathbf{v}}_i^{n+1} \rightarrow \tilde{\mathbf{v}}_i^{n+1}$. The collision processing is described in (§8).

5.3 Transfer to particles

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

$$\mathbf{v}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1}. \qquad (11)$$

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

$$\mathbf{B}_p^{n+1} = \sum_i w_{ip}^n \tilde{\mathbf{v}}_i^{n+1} (\mathbf{x}_i^n - \mathbf{x}_p^n)^T. \qquad (12)$$

5.4 Update particle state

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

$$\mathbf{x}_p^{n+1} = \sum_i w_{ip}^n \overline{\mathbf{x}}_i^{n+1}. \qquad (13)$$

Since the particles move with the flow, the material derivative in Equation 3 is just a normal time derivative and a simple difference yields the particle deformation gradient update

$$\mathbf{F}_p^{n+1} = \mathbf{F}_p^n + \Delta t (\nabla \mathbf{v})_p \mathbf{F}_p^n \qquad (14)$$

where $(\nabla \mathbf{v})_p$ is calculated by differentiating (11)

$$(\nabla \mathbf{v})_p = \sum_i \overline{\mathbf{v}}_i^{n+1} (\nabla w_{ip}^n)^T. \qquad (15)$$

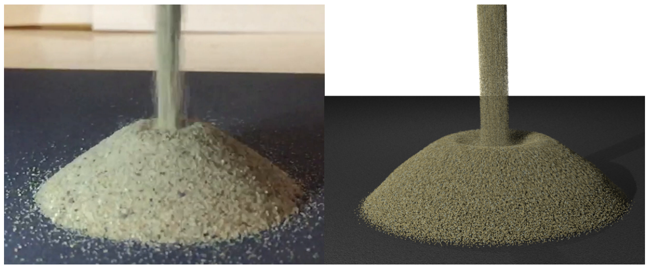

Figure 12: *Sand is poured from a spout into a pile in a lab (left) and with our method (right).*

Note that we only store the elastic ($\mathbf{F}_p^{E,n}$) and plastic ($\mathbf{F}_p^{P,n}$) parts of $\mathbf{F}_p^n$ rather than $\mathbf{F}_p^n$ itself. These are related by $\mathbf{F}_p^n = \mathbf{F}_p^{E,n} \mathbf{F}_p^{P,n}$ (§7). During this evolution step, we assume that the plastic part is not changing ($\hat{\mathbf{F}}_p^{n+1} = \mathbf{F}_p^n$), which gives us the rule

$$\hat{\mathbf{F}}_p^{E,n+1} = \mathbf{F}_p^{E,n} + \Delta t (\nabla \mathbf{v})_p \mathbf{F}_p^{E,n}. \qquad (16)$$

The plastic update is covered in (§7).

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

5.5 Plasticity, hardening

The final step is to apply plasticity and hardening. Plasticity is performed by projecting the elastic deformation gradient to its yield surface, an action denoted by $\mathbf{Z}(\cdot, \cdot)$ which we describe in detail later (§7). Plasticity does not change the full deformation gradient, so that $\mathbf{F}_p^{n+1} = \hat{\mathbf{F}}_p^{E,n+1} \hat{\mathbf{F}}_p^{P,n+1} = \mathbf{F}_p^{E,n+1} \mathbf{F}_p^{P,n+1}$. This allows us to update the plastic part.

$$\mathbf{F}_p^{E,n+1} = \mathbf{Z}(\hat{\mathbf{F}}_p^{E,n+1}, \alpha_p^n) \qquad (17)$$

$$\mathbf{F}_p^{P,n+1} = (\mathbf{F}_p^{E,n+1})^{-1} \hat{\mathbf{F}}_p^{E,n+1} \hat{\mathbf{F}}_p^{P,n+1} \qquad (18)$$

The last step is hardening which updates $\alpha_p^n \rightarrow \alpha_p^{n+1}$ (§7.3).

5.6 Implicit velocity update

The implicit velocity update is

$$\overline{\mathbf{v}}_i^{n+1} = \mathbf{v}_i^n + \frac{\Delta t}{m_i^n} \mathbf{f}_i(\langle \mathbf{F}_p^{E,n+1} \rangle) + \sum_k \nabla G_{ki} \lambda_k \qquad (19)$$

subject to the additional conditions $G_k \geq 0$, $\lambda_k \geq 0$, and $G_k \lambda_k = 0$. Here, $G_k(\langle \overline{\mathbf{x}}_i^{n+1} \rangle) \geq 0$, with $\overline{\mathbf{x}}_i^{n+1} = \mathbf{x}_i^n + \Delta t \overline{\mathbf{v}}_i^{n+1}$, is the collision-free criterion for all object-node collision pairs k. (§8) These forces are implicit, since $\mathbf{F}_p^{E,n+1}$ depends on $\overline{\mathbf{v}}_i^{n+1}$ through (16), (15), and (17). As in the explicit case, we complete the grid update by applying friction $\overline{\mathbf{v}}_i^{n+1} \rightarrow \tilde{\mathbf{v}}_i^{n+1}$ and described in (§8).

Note that we are implicit in plasticity, but we are not implicit in hardening or friction. In the absence of plasticity, these are just the Karush-Kuhn-Tucker (KKT) conditions [Nocedal and Wright 2006] for minimization. Unlike solving a minimization problem, however, our linear systems are not generally symmetric, and we do not have an objective with which to do line searches.

Solving the system. Since the collision constraints are independent, we use the projection method to eliminate the collisions. We

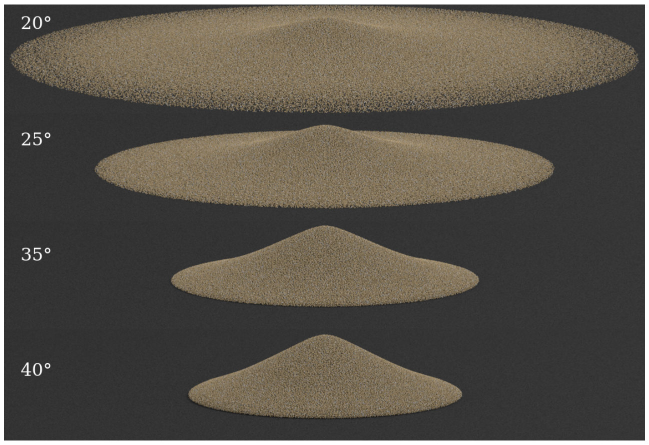

Figure 13: *Varying the friction angle changes the shape of a pile of sand. A larger angle produces a taller sand pile with steeper sides.*

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

5.7 Initialization

Particle locations are initialized with Poisson disk sampling. Initial values for m_p, $\mathbf{x}_p^n$, and $\mathbf{v}_p^n = \mathbf{v}(\mathbf{x}_p^n)$ are chosen based on the needs of the example, with $\mathbf{v}(\mathbf{x})$ the desired initial velocity field. $\mathbf{B}_p^n$ is initialized so that $\mathbf{C}_p^n = \mathbf{B}_p^n(\mathbf{D}_p^n)^{-1} = \nabla\mathbf{v}$ is the gradient of the initial velocity field and $\mathbf{D}_p^n$ is computed from (8). Our initial setups have no deformation, so $\mathbf{F}_p^{E,n} = \mathbf{F}_p^{E,n+1} = \mathbf{I}$. We initialize our hardening parameter with $q_p^n = 0$, from which we can compute α_p^n using (30) and (31). Initial particle volume V_p^0 is computed from the seeding density.

6 Forces

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

6.1 Continuous setting

Elastic materials are characterized by their ability to store potential energy and then release it by doing work to cause motion (kinetic energy). Let Ψ be the total potential energy stored by a material at a given time. In a real material, potential energy is stored locally in response to deformation. This is called *energy density*, or energy per unit volume, and represented by ψ. Since this depends only on the local deformation, we can write $\psi(\mathbf{F})$. The function $\psi(\mathbf{F})$ captures the essential information about the way an elastic material responds to deformation. This relationship depends on the material; we choose our model in (§6.3).

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

6.2 Discrete setting

We discretize the potential energy with a sum on particles,

$$\Psi = \sum_p V_p^0 \psi(\mathbf{F}_p^E) \tag{20}$$

Note that V_p^0 is the volume of material attributed to a particle in the initial configuration. Only the elastic portion of the deformation gradient $\mathbf{F}_p^E$ contributes to the energy [Bonet and Wood 2008].

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

With MPM, grid nodes are temporarily Lagrangian, and can be moved to define the force. If the current grid node velocity is $\mathbf{v}_i$, then its position can be approximated as $\mathbf{x}_i = \mathbf{x}_i^n + \Delta t\mathbf{v}_i$. Considering a different ending position implies a different velocity to get there. These node velocities are used in (15) to compute $(\nabla\mathbf{v})_p$, which is in turn used by (16) to compute a new deformation gradient $\mathbf{F}_p^E$. This deformation gradient will be used to compute energy density using a model $\psi(\mathbf{F}_p^E)$, which finally gives us total potential energy. In this way, the potential energy of the material can be expressed in terms of the locations of the grid nodes. We can use this relationship, summarized below, to compute forces on grid nodes.

$$\Psi(\langle\mathbf{x}_i\rangle) = \sum_p V_p^0 \psi(\mathbf{F}_p^E(\langle\mathbf{x}_i\rangle)) \tag{21}$$

$$\mathbf{F}_p^E(\langle\mathbf{x}_i\rangle) = \left(\mathbf{I} + \sum_i (\mathbf{x}_i - \mathbf{x}_i^n)(\nabla w_{ip}^n)^T\right)\mathbf{F}_p^{E,n} \tag{22}$$

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

$$\mathbf{f}_i(\langle\mathbf{F}_p^E\rangle) = -\frac{\partial\Psi}{\partial\mathbf{x}_i} = -\sum_p V_p^0 \left(\frac{\partial\psi}{\partial\mathbf{F}}(\mathbf{F}_p^E)\right)(\mathbf{F}_p^{E,n})^T \nabla w_{ip}^n. \tag{23}$$

Note that $\mathbf{F}_p^E$ is the function parameter but $\mathbf{F}_p^{E,n}$ is a known value which is not changing during the current time step. Also note that all deformation is assumed to be elastic. When computing the force, the effect of further plastic flow is ignored [Bonet and Wood 2008].

Gravity. We include gravity as an additional term in (23)

$$\mathbf{f}_i^{grav} = \sum_p w_{ip}^n m_p \mathbf{g} = m_i^n \mathbf{g}. \tag{24}$$

6.3 Constitutive model

We adopt the energy density $\psi(\mathbf{F})$ from Mast et al. [2013]. This model uses the same energy density as St. Venant-Kirchhoff, but it replaces the left Cauchy Green strain with the Hencky strain $\frac{1}{2}\ln(\mathbf{F}\mathbf{F}^T)$. This makes a number of aspects of the Drucker-Prager plastic projection very simple (see the supplementary technical document [Klár et al. 2016]). The model is most conveniently written in terms of the singular value decomposition $\mathbf{F} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ as

$$\psi(\mathbf{F}) = \mu\operatorname{tr}\big((\ln\mathbf{\Sigma})^2\big) + \frac{1}{2}\lambda(\operatorname{tr}(\ln\mathbf{\Sigma}))^2, \tag{25}$$

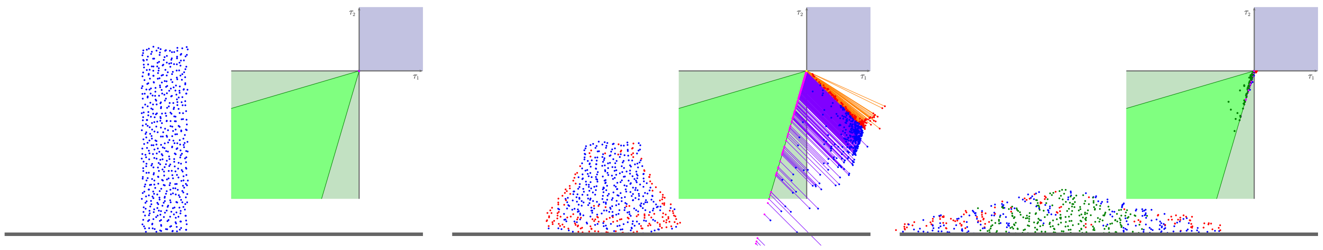

Figure 14: *A column of sand collapses into a pile. Sand particles are colored based on their current plastic deformation behavior. The plot shows the locations of these particles in principal stress space. Green particles lie within the yield surface and experience no plasticity. Blue particles are projected to the yield surface along a direction that avoids volume change. Red particles are experiencing tension and are projected to the tip of the conical yield surface; these particles are separating freely with no stress.*

where $\mathbf{\Sigma}$ is diagonal so $\ln \mathbf{\Sigma}$ is computed by taking the logarithm of the diagonal entries. The force computation (23) requires the derivative of this, which is

$$\frac{\partial \psi}{\partial \mathbf{F}}(\mathbf{F}) = \mathbf{U}(2\mu\mathbf{\Sigma}^{-1}\ln\mathbf{\Sigma} + \lambda \mathrm{tr}(\ln\mathbf{\Sigma})\mathbf{\Sigma}^{-1})\mathbf{V}^T. \qquad (26)$$

7 Plasticity

7.1 Projecting to the yield surface

The only algorithmic aspect of our plasticity treatment that has not yet been defined is our function $\mathbf{Z}(\mathbf{F}_p^E, \alpha_p)$, which projects the deformation gradient $\mathbf{F}_p^E$ to the yield surface defined by the parameter α_p. The Drucker-Prager plasticity model is based on Coulomb friction interactions between sand particles. In the continuum setting, this means that the shear stress cannot be larger than a coefficient of friction times the normal stress. This results in a simple constraint on the principal stresses which we derive in the supplementary technical document [Klár et al. 2016].

In the space of principal stress, the yield surface looks like a cone (see Figure 15). There are three possible cases that must be considered. If the stress lies within the yield surface (Case I), then there is static friction between sand particles, and no plasticity occurs. If the sand is undergoing expansion (Case II), then there is no resistance to motion; this corresponds to the tip of the cone. Otherwise, there is dynamic friction (Case III), and we should project to the side of the cone. Examples of these cases in an actual simulation can be seen in Figure 14.

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

$$\hat{\boldsymbol{\epsilon}}_p = \boldsymbol{\epsilon}_p - \frac{\mathrm{tr}(\boldsymbol{\epsilon}_p)}{d}\mathbf{I} \quad \delta\gamma_p = \|\hat{\boldsymbol{\epsilon}}_p\|_F + \frac{d\lambda + 2\mu}{2\mu}\mathrm{tr}(\boldsymbol{\epsilon}_p)\alpha_p \qquad (27)$$

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

$$\mathbf{H}_p = \boldsymbol{\epsilon}_p - \delta\gamma_p \frac{\hat{\boldsymbol{\epsilon}}_p}{\|\hat{\boldsymbol{\epsilon}}_p\|_F} \qquad (28)$$

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

7.2 Note on preventing undesired volume change

The method as described has the desirable feature that sand is prevented from compressing arbitrarily as a byproduct of losing volume in the plasticity projection. To see this, note that a change in $\det(\mathbf{F}_p^E)$ corresponds to a change in volume of the elastic deformation. In Case I, $\mathbf{F}_p^E$ is unchanged, so volume is not changed. In Case II, the sand expands, and volume should be gained. In Case III, the sand deforms plastically and an associative flow rule [Bonet and

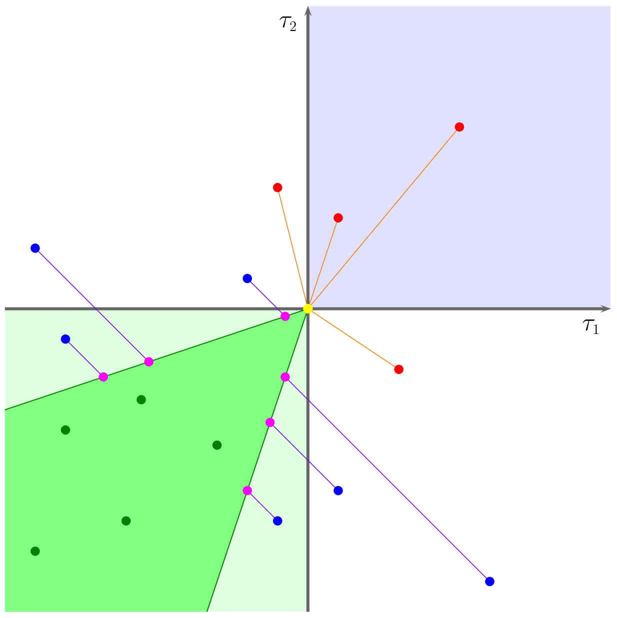

Figure 15: *The yield surface for Drucker-Prager is shown in principle stretch space. The yield surface has the shape of a cone with its tip at the origin, which corresponds to no stress. Green particles are inside the yield surface and exhibit an elastic response. Blue particles are under compression but experience more shear than friction allows. These configurations are projected to the yield surface along a direction that avoids volume change. Red particles are experiencing tension and are projected to the tip of the conical yield surface. These particles separate freely without stress.*

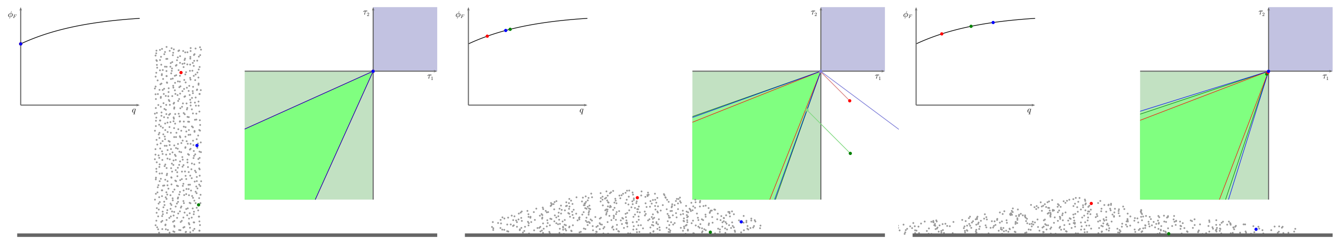
Figure 16: *Three particles in a collapsing pile of sand are colored for reference. As these particles deform plasticly, their yield surface changes as they undergo hardening, resulting in a wider cone for projection. Hardening causes each particle to have its own yield surface.*

Wood 2008] would lead to excessive volume gain. Instead, noting that $\text{tr}(\hat{\boldsymbol{\epsilon}}_p) = 0$, the Drucker-Prager model uses a non-associative flow to preserve volume during the plastic projection

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

The key to retaining volume in this case is to ensure that $\text{tr}(\mathbf{H}_p) = \text{tr}(\boldsymbol{\epsilon}_p)$, which means the projection to the cone should locate the closest point on the cone that does not change the trace, rather than the closest point on the cone. We discuss this in more detail in the supplementary technical document [Klár et al. 2016].

7.3 Hardening

We adopt the hardening model of Mast et al. [2014], where plastic deformation can increase the friction between sand particles. The amount of hardening depends on the amount of correction that occurred due to plasticity. In Case I, no plasticity occurred, so $\delta q_p = 0$. In Case II, all of the stress was removed, so $\delta q_p = \|\boldsymbol{\epsilon}_p^{E,n+1}\|_F$. In Case III, the amount of plasticity that occurred was $\delta q_p = \delta\gamma_p$. In each case, $\delta q_p \geq 0$. We define our hardening update using

$$q_p^{n+1} = q_p^n + \delta q_p \tag{29}$$

$$\phi_{F\,p} = h_0 + (h_1 q_p^{n+1} - h_3) e^{-h_2 q_p^{n+1}} \tag{30}$$

$$\alpha_p^{n+1} = \sqrt{\frac{2}{3}} \frac{2 \sin \phi_{F\,p}}{3 - \sin \phi_{F\,p}} \tag{31}$$

The quantity q_p^n is the hardening state, $\phi_{F\,p}$ is often referred to as the friction angle, the *internal coefficient of friction* is $\tan \phi_{F\,p}$, and (30) models a curve with a maximum and an asymptote. Plausible values of $\phi_{F\,p}$ lie in $[0, \frac{\pi}{2})$, with $\phi_{F\,p} = 0$ behaving as a fluid. Feasible hardening parameters satisfy $h_0 > h_3 \geq 0$ and $h_1, h_2 \geq 0$. The values we use are listed in Table 3. Figure 16 illustrates the change in yield surface as particles undergo hardening in 2D.

8 Collisions

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

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

Since $\mathbf{x}_p^{n+1}$ will be computed based on $\overline{\mathbf{v}}_i^{n+1}$, one could adjust $\overline{\mathbf{v}}_i^{n+1}$ to enforce $\phi(\mathbf{x}_p^{n+1}) \geq 0$. While this would likely lead to good results, it complicates collision processing in both the explicit and implicit cases. Instead, we process collisions against the nodes themselves as in [Gast et al. 2015]. This is difficult because $\phi(\overline{\mathbf{x}}_i^{n+1}) \geq 0$ does not make sense. There *should* be grid nodes inside obstacles. A set of constraints $G_k(\langle\overline{\mathbf{x}}_i^{n+1}\rangle) \geq 0$ or $G_k(\langle\overline{\mathbf{x}}_i^{n+1}\rangle) = 0$ on grid nodes is needed that avoid collisions for particles, at least approximately, but which can be applied independently per grid node in an straightforward manner. This depends on the type of collision being applied. We support three types of collisions: sticky, slipping, and separating. Note that a grid node must have received mass during the transfer in order to be considered for a collision constraint of any type.

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

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

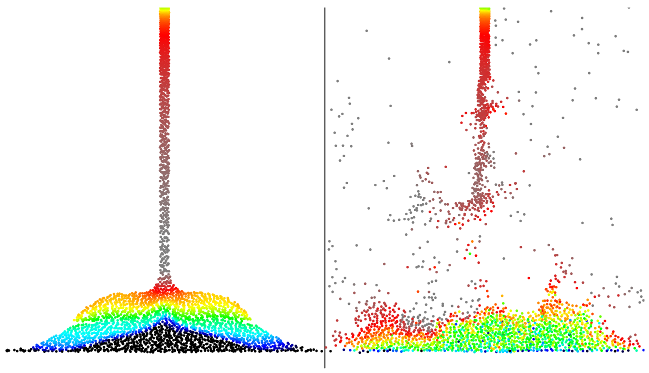
Figure 17: *Sand is poured into a pile with APIC (left) and FLIP (right) transfers. FLIP tends to accumulate spurious velocities on particles. In some cases, FLIP leads to unstable behavior, as was the case in this simulation.*

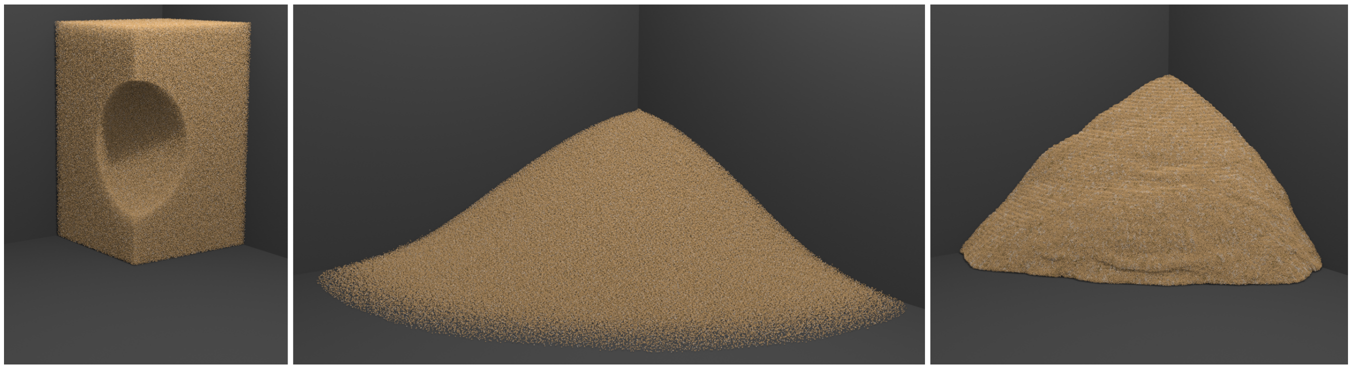

Figure 18: *Comparison on notched sand block fall. Initial (left), ours (middle), and Narain et al. [2010] (right).*

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

With a mathematical description for the constraints for all cases and a method for directly enforcing those constraints, direct enforcement ($\mathbf{v}_i^\star \rightarrow \overline{\mathbf{v}}_i^{n+1}$) is all that is required for the explicit case. The implicit case uses the constraints G_k that have been defined in order to couple collision enforcement with force application (§5.6).

8.1 Friction

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

In the implicit case, the collision contribution is from the last term of Equation 19. We compute the velocity estimate before forces as $\mathbf{v}_i^\star = \mathbf{v}_i^n + \frac{\Delta t}{m_i^n} \mathbf{f}_i(\langle \mathbf{F}_p^{E,n+1} \rangle)$. Although $\overline{\mathbf{v}}_i^{n+1}$ would be the after-collision velocity if the implicit solve had converged, this is not often done in practice. Instead, we repeat collision processing on $\mathbf{v}_i^\star$ to compute the difference for $\Delta \mathbf{v}_i$.

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

9 Render

Zhu and Bridson [2005] render sand with a reconstructed surface. Narain et al. [2010] associate a number of render points sampled near each particle for high resolution rendering. In our examples we have a sufficient number of simulated particles to simply render each particle as a matte sphere. The color of each particle is randomly chosen from yellow (RGB $225/169/95$ with a probability of 0.85), brown (RGB $107/84/30$ with a probability of 0.1) and white (RGB $255/255/255$ with a probability of 0.05) to further improve realism. All scenes were rendered using SideFX's Mantra. For scenes with rapidly flowing sand (such as the hourglass) we turned on motion blur where appropriate.

10 Results

Flowing and Piling. We demonstrate the accuracy of our model by showing the characteristic behaviors of sand flowing and piling. In Figure 1, we simulate sand flowing inside an hourglass. The sand forms a smooth granular flow and piles up at the bottom. Figure 12 shows a stream of sand inflow hitting a high frictional surface. We compare this simulation with real world footage. Our model successfully captures the interesting avalanche instability [Yoshioka 2003] of this experiment.

Easy Tuning. We list the parameters used in our examples in Table 3 and the runtime performance of those simulations in Table 2. In Figure 13, we simulate columns of dry sand with different friction angles collapsing on the ground. Different friction angles directly affect the interaction between sand grains, therefore the final piling angle. While the real Young's modulus of sand is 3.537×10^7, we found that sometimes choosing a moderately smaller value does not change the visual appearance. In Figure 10, we show 2D inflow simulations with different Young's modulus. A moderately smaller Young's modulus improves the efficiency of the implicit solve. However, the material may exhibit jiggling behavior if it is too small. We assert physically accurate Young's modulus is always the best choice unless an artistic elastic effect is desirable.

Two-way Coupling. The benefits of using MPM include automatic self collision and coupling between different materials. In Figure 3, we show an elastic ball interacting with a dry sand castle. MPM naturally handles the two-way coupling without requiring any additional treatment other than assigning different constitutive models to different particles.

Drawing and Scooping. We further demonstrate the versatility of our method by performing various tasks in a sand box. Figure 8 shows drawing a butterfly with a wooden stick. Figure 5 shows raking sand in a Zen garden. Figure 11 shows scooping sand.

APIC Stability. Our method benefits from the APIC particle/grid transfers [Jiang 2015; Jiang et al. 2015] due to its stability and low numerical dissipation. In Figure 17, we run a 2D inflow simulation and compare our result with Mast et al. [2013; 2014] where a traditional FLIP transfer scheme is used. With the same material parameters and time step sizes, our method does not suffer from the

	Frame rate	Min/frame	Scheme	Δt	Particle #	Threads	CPU	Δx	Grid resolution	Particles/cell
Castle	72	6.80	Implicit	1×10^{-3}	7.95×10^{5}	4	†2.67GHz	0.01	$300 \times 140 \times 200$	8
Friction angle	120	4.10	Explicit	1×10^{-4}	1.20×10^{6}	4	‡3.33GHz	0.001	$432 \times 144 \times 432$	32
Hourglass	48	2.00	Explicit	1×10^{-4}	4.60×10^{5}	12	*3.00GHz	0.0025	$160 \times 360 \times 160$	8
Butterfly	24	10.31	Explicit	2.5×10^{-4}	3.84×10^{6}	10	*3.00GHz	0.0045	$220 \times 55 \times 220$	8
Butterfly close	48	21.23	Explicit	1×10^{-4}	4.1×10^{6}	8	*3.00GHz	0.0014	$280 \times 140 \times 210$	2
Raking	24	9.78	Explicit	2.5×10^{-4}	3.84×10^{6}	10	*3.00GHz	0.0045	$220 \times 55 \times 220$	8
Raking close	24	32.84	Explicit	1×10^{-4}	4.3×10^{6}	8	$^{\circ}$2.90GHz	0.0021	$336 \times 96 \times 288$	2
Pile from spout	120	4.34	Implicit	1.5×10^{-4}	9.94×10^{5}	8	†2.67GHz	0.00083	$240 \times 96 \times 240$	32
Splash	240	9.27	Explicit	5×10^{-5}	6.6×10^{6}	8	§3.47GHz	0.0078	$256 \times 256 \times 256$	9
Shovel	24	24.84	Explicit	1×10^{-4}	1.96×10^{6}	4	§3.47GHz	0.005	$160 \times 100 \times 100$	8
Young's modulus	24	49×10^{-4}	Implicit	7.5×10^{-4}	742/739/746/745	1	†2.67GHz	0.016	256×64	4

Table 2: *Simulation performance. Note that Δt denotes the maximum allowed time step size. The actual Δt is adaptive and may be restricted by CFL condition when the particle velocities are high. In all of our simulations we use a CFL number of 1, i.e., we don't allow particles to move further than Δx in a time step. CPU types used are: †Intel Xeon X5650, ‡Intel Xeon W3680, *Intel Xeon E5-2690 v2, $^{\circ}$Intel Xeon E5-2690, §Intel Xeon X5690.*

unstable ringing instability like FLIP does. We further show the robustness and stability of our method in a 3D energetic scenario. In Figure 2, a rigid ball is dropped into a $1m \times 1m \times 0.35m$ sand box with impact speed $6m/s$. The impact dynamics are stable and almost noise-free, resulting in a smooth and symmetric crown splash visual appearance.

Comparison with the state-of-the-art method. We compare the result of our method with the algorithm proposed by Narain et al. [2010] for the collapse of a column of granular material. The results shown in Figure 18 are at $150 \times 100 \times 150$ grid resolution. Performance data for this example at a variety of resolutions is shown in Table 4. Two particle counts (initial and final) are listed for Narain et al., since their particle counts tend to increase over time due to particle splitting and merging. Although their algorithm runs 8.7 times faster at the highest resolution, our algorithm avoids the staircasing artifact and is able to produce a less viscous flow of dry cohesion-less granular materials.

11 Discussion and Limitations

Limitations. There are methods that are much faster, for example the position based dynamics approach in Macklin et al. [2014] or other existing continuum approaches such as Narain et al. [2010]. However, when realism and intuitively designed parameters are more important than raw performance, our method provides an alternative with competitive computational expense. Also, although the framework would generalize to a wide range of yield surfaces and elastic potentials, we only investigated the Drucker-Prager model. However, the Drucker-Prager cone is only equivalent to the Coulomb friction shear/normal-stress relation in two dimensions. In three dimensions, the elastic regime is described by the more complicated region in the Mohr-Coulomb model, but the Drucker-Prager model is a decent approximation [Mast 2013].

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

Discussion. We note that explicit time stepping was often faster than implicit time stepping. Although implicit steps are generally larger than explicit, the cost required to solve the nonlinear equations of the implicit step was often larger than just taking more inexpensive explicit steps. Improvements in stability of explicit integration may be partly due to a better position update and APIC transfers providing more stability than FLIP/PIC blends as in Stomakhin et al. [2013], whose implicit scheme also benefited from a symmetric treatment. Tuning time step and solver tolerances proved difficult to optimize, requiring different values for different examples.

Acknowledgements

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

References

ALDUÁN, I., AND OTADUY, M. 2011. SPH granular flow with friction and cohesion. In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, 25–32.

ALDUÁN, I., TENA, A., AND OTADUY, M. 2009. Simulation of high-resolution granular media. In *Proc Cong Español Inf Graf*.

BARDENHAGEN, S., BRACKBILL, J., AND SULSKY, D. 2000. The material-point method for granular materials. *Comp Meth App Mech Eng 187*, 3–4, 529–541.

BELL, N., YU, Y., AND MUCHA, P. 2005. Particle-based simulation of granular materials. In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, 77–86.

BONET, J., AND WOOD, R. 2008. *Nonlinear continuum mechanics for finite element analysis*. Cambridge University Press.

CHANCLOU, B., LUCIANI, A., AND HABIBI, A. 1996. Physical models of loose soils dynamically marked by a moving object. In *Comp Anim*, 27–35.

CHANG, Y., BAO, K., ZHU, J., AND WU, E. 2012. A particle-based method for granular flow simulation. *Sci China Inf Sci 55*, 5, 1062–1072.

	ρ	E	ν	Friction angle	$h_0/h_1/h_2/h_3$
Castle	2200	3.537×10^{5}	0.3	—	35/0/0.2/10
Friction angle	2200	3.537×10^{5}	0.3	20/25/30/35/40	—
Hourglass	2200	3.537×10^{5}	0.3	—	35/9/0.3/10
Butterfly	2200	3.537×10^{5}	0.3	—	35/9/0.2/10
Butterfly close	2200	3.537×10^{5}	0.3	—	35/9/0.2/10
Raking	2200	3.537×10^{5}	0.3	—	35/9/0.2/10
Raking close	2200	3.537×10^{5}	0.3	—	35/9/0.2/10
Pile from spout	2200	3.537×10^{5}	0.3	30	—
Splash	1582	3.537×10^{6}	0.3	22	—
Shovel	2200	3.537×10^{5}	0.3	—	35/9/0.2/10
Young's modulus	2200	$10^{3,4,5,6}$	0.3	—	35/9/0.3/10

Table 3: *Material parameters are provided for all of our 3D simulations. Friction angle ϕ_F and hardening parameters h_0, h_1, and h_3 are listed in degrees for convenience.*

CHEN, P., AND WONG, S. 2013. Real-time auto stylized sand art drawing. In *CAD Comp Graph*, 439–440.

CUMMINS, S., AND BRACKBILL, J. 2002. An implicit particle-in-cell method for granular materials. *J Comp Phys 180*, 2, 506–548.

DRUCKER, D., AND PRAGER, W. 1952. Soil mechanics and plasticity analysis or limit design. *Quart App Math 10*, 157–165.

GAST, T., SCHROEDER, C., STOMAKHIN, A., JIANG, C., AND TERAN, J. 2015. Optimization integrator for large time steps. *IEEE Trans Vis Comp Graph 21*, 10, 1103–1115.

GONZALEZ, O., AND STUART, A. 2008. *A first course in continuum mechanics*. Cambridge University Press.

IHMSEN, M., WAHL, A., AND TESCHNER, M. 2013. A lagrangian framework for simulating granular material with high detail. *Comp Graph 37*, 7, 800–808.

JAEGER, H., NAGEL, S., AND BEHRINGER, R. 1996. Granular solids, liquids, and gases. *Rev Mod Phys 68*, 1259–1273.

JIANG, C., SCHROEDER, C., SELLE, A., TERAN, J., AND STOMAKHIN, A. 2015. The affine particle-in-cell method. *ACM Trans Graph 34*, 4, 51:1–51:10.

JIANG, C. 2015. *The material point method for the physics-based simulation of solids and fluids*. PhD thesis, University of California, Los Angeles.

KLÁR, G., GAST, T., PRADHANA, A., FU, C., SCHROEDER, C., JIANG, C., AND TERAN, J. 2016. Drucker-prager elastoplasticity for sand animation: Supplementary technical document. *ACM Trans Graph.*

LENAERTS, T., AND DUTRÉ, P. 2009. Mixing fluids and granular materials. *Comp Graph Forum 28*, 2, 213–218.

LI, X., AND MOSHELL, J. 1993. Modeling soil: realtime dynamic models for soil slippage and manipulation. In *Proc SIGGRAPH*, 361–368.

LUCIANI, A., HABIBI, A., AND MANZOTTI, E. 1995. A multi-scale physical model of granular materials. In *Proc Graph Int*, 136–146.

MACKLIN, M., MÜLLER, M., CHENTANEZ, N., AND KIM, T. 2014. Unified particle physics for real-time applications. *ACM Trans Graph 33*, 4, 153:1–153:12.

MAST, C., ARDUINO, P., MACKENZIE-HELNWEIN, P., AND MILLER, R. 2014. Simulating granular column collapse using the material point method. *Acta Geotech 10*, 1, 101–116.

MAST, C. 2013. *Modeling landslide-induced flow interactions with structures using the Material Point Method.* PhD thesis.

MAZHAR, H., HEYN, T., NEGRUT, D., AND TASORA, A. 2015. Using Nesterov's method to accelerate multibody dynamics with friction and contact. *ACM Trans Graph 34*, 3, 32:1–32:14.

MILENKOVIC, V. 1996. Position-based physics: simulating the motion of many highly interacting spheres and polyhedra. In *Proc SIGGRAPH*, 129–136.

MILLER, G., AND PEARCE, A. 1989. Globular dynamics: a connected particle system for animating viscous fluids. *Comp Graph 13*, 3, 305–309.

NARAIN, R., GOLAS, A., AND LIN, M. 2010. Free-flowing granular materials with two-way solid coupling. *ACM Trans Graph 29*, 6, 173:1–173:10.

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

Table 4: *Performance comparison with Narain et al. [2010].*

NKULIKIYIMFURA, D., KIM, J., AND KIM, H. 2012. A real-time sand simulation using a GPU. In *Comp Tech Inf Man*, vol. 1, 495–498.

NOCEDAL, J., AND WRIGHT, S. 2006. *Numerical Optimization*. Springer series in operations research and financial engineering. Springer.

ONOUE, K., AND NISHITA, T. 2003. Virtual sandbox. In *Proc Pac Conf Comp Graph App*, 252–262.

PLA-CASTELLS, M., GARCIA-FERNANDEZ, I., AND MARTINEZ, R. 2006. Interactive terrain simulation and force distribution models in sand piles. In *Cellular Automata*, vol. 4173 of *Lecture Notes Comp Sci*. 392–401.

RAM, D., GAST, T., JIANG, C., SCHROEDER, C., STOMAKHIN, A., TERAN, J., AND KAVEHPOUR, P. 2015. A material point method for viscoelastic fluids, foams and sponges. In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, 157–163.

STEFFEN, M., KIRBY, R. M., AND BERZINS, M. 2008. Analysis and reduction of quadrature errors in the material point method (MPM). *Int J Numer Meth Eng 76*, 6, 922–948.

STOMAKHIN, A., SCHROEDER, C., CHAI, L., TERAN, J., AND SELLE, A. 2013. A material point method for snow simulation. *ACM Trans Graph 32*, 4, 102:1–102:10.

STOMAKHIN, A., SCHROEDER, C., JIANG, C., CHAI, L., TERAN, J., AND SELLE, A. 2014. Augmented MPM for phase-change and varied materials. *ACM Trans Graph 33*, 4, 138:1–138:11.

SULSKY, D., CHEN, Z., AND SCHREYER, H. L. 1994. A particle method for history-dependent materials. *Comp Meth in App Mech Eng 118*, 1, 179–196.

SUMNER, R., O'BRIEN, J., AND HODGINS, J. 1999. Animating sand, mud and snow. *Comp Graph Forum 18*, 1, 17–26.

YASUDA, R., HARADA, T., AND KAWAGUCHI, Y. 2008. Real-time simulation of granular materials using graphics hardware. In *Comp Graph Imag Vis*, 28–31.

YOSHIOKA, N. 2003. A sandpile experiment and its implications for self-organized criticality and characteristic earthquake. *Earth, planets and space 55*, 6, 283–289.

YUE, Y., SMITH, B., BATTY, C., ZHENG, C., AND GRINSPUN, E. 2015. Continuum foam: a material point method for shear-dependent flows. *ACM Trans Graph 34*, 5, 160:1–160:20.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. *ACM Trans Graph 24*, 3, 965–972.