Augmented MPM for phase-change and varied materials

Alexey Stomakhin* Craig Schroeder† Chenfanfu Jiang† Lawrence Chai* Joseph Teran*† Andrew Selle*

†University of California Los Angeles *Walt Disney Animation Studios

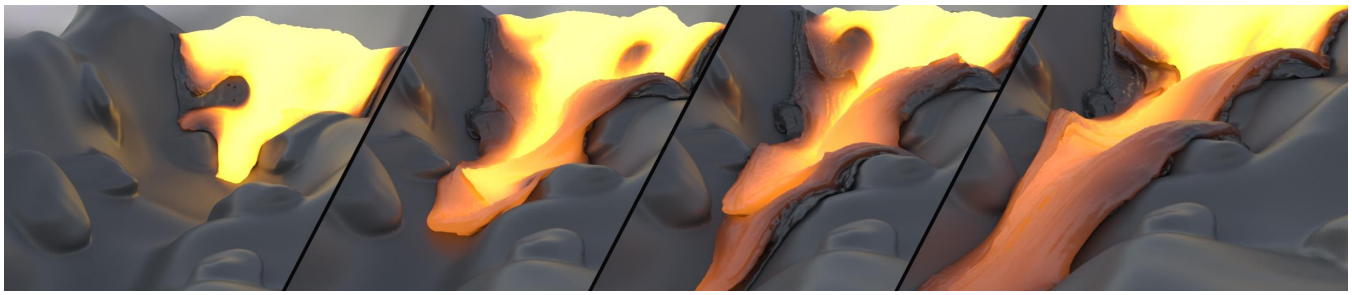

Figure 1: *Lava solidifying into pāhoehoe forms complex and attractive shapes. The lava emits light according to the blackbody spectrum corresponding to the simulated temperature. ©Disney.*

Abstract

In this paper, we introduce a novel material point method for heat transport, melting and solidifying materials. This brings a wider range of material behaviors into reach of the already versatile material point method. This is in contrast to best-of-breed fluid, solid or rigid body solvers that are difficult to adapt to a wide range of materials. Extending the material point method requires several contributions. We introduce a dilational/deviatoric splitting of the constitutive model and show that an implicit treatment of the Eulerian evolution of the dilational part can be used to simulate arbitrarily incompressible materials. Furthermore, we show that this treatment reduces to a parabolic equation for moderate compressibility and an elliptic, Chorin-style projection at the incompressible limit. Since projections are naturally done on marker and cell (MAC) grids, we devise a staggered grid MPM method. Lastly, to generate varying material parameters, we adapt a heat-equation solver to a material point framework.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: material point, lava, freezing, melting, physically-based modeling

Links: DL PDF WEB

1 Introduction

From the process of lava solidifying into *pāhoehoe* to advertisements showing caramel and chocolate melting over ice cream, materials undergoing phase transitions are both ubiquitous and complex. These transitional dynamics are some of the most compelling natural phenomena. However, visual simulation of these effects remains a challenging open problem. The difficulty lies in achieving robust, accurate and efficient simulation of a wide variety of material behaviors without requiring overly complex implementations.

Phase transitions and multiple material interactions typically involve large deformation and topological changes. Thus a common approach is to use a modified fluid solver, which works well for viscous Newtonian-fluids or even moderately viscoplastic flows. However, solid and elastic material behavior is then more difficult to achieve. Alternatively, Lagrangian-mesh-based approaches naturally resolve elastic deformation, but they must be augmented with explicit collision detection, and remeshing is required for fluid-like behaviors with topological changes. Due to the tradeoffs between solid and fluid methods, many authors have considered explicit coupling between two solvers, but such approaches typically require complex implementations and have significant computational cost.

A common goal is to handle a variety of materials and material transitions without sacrificing simplicity of implementation. This motivation typically drives researchers and practitioners toward particle approaches. For example, SPH and FLIP methods are commonly augmented with an approach for computing strains required for more general elastic stress response. The key observation is that particles are a simple and extremely flexible representation for graphics. This is a central motivation in our approach to the problem.

Computing strain from world-space particle locations without the luxury of a Lagrangian mesh proves challenging. One approach is using the material point method (MPM) [Sulsky et al. 1995], which augments particles with a transient Eulerian grid that is adept at computing derivatives and other quantities. However, while MPM methods successfully resolve a wide range of behaviors, they do not handle arbitrarily incompressible materials. This is in contrast to incompressible FLIP [Zhu and Bridson 2005] techniques that naturally treat liquid simulation but typically only resolve pressure or viscosity-based stresses.

In this paper, we present a number of contributions. We show that MPM can be easily augmented with a Chorin-style projection [Chorin 1968] technique that enables simulation of arbitrarily incompressible materials thus providing a connection to the commonly used FLIP techniques. We achieve this with a MAC-grid-based [Harlow and Welch 1965] MPM solver, a splitting of the stress into elastic and dilational parts, a projection-like implicit

ACM Reference Format
Stomakhin, A., Schroeder, C., Jiang, C., Chai, L., Teran, J., Selle, A. 2014. Augmented MPM for phase-change and varied materials. ACM Trans. Graph. 33, 4, Article 138 (July 2014), 11 pages.
DOI = 10.1145/2601097.2601176 http://doi.acm.org/10.1145/2601097.2601176.

0730-0301/14/07-ART138 $15.00.
DOI: http://dx.doi.org/10.1145/2601097.2601176

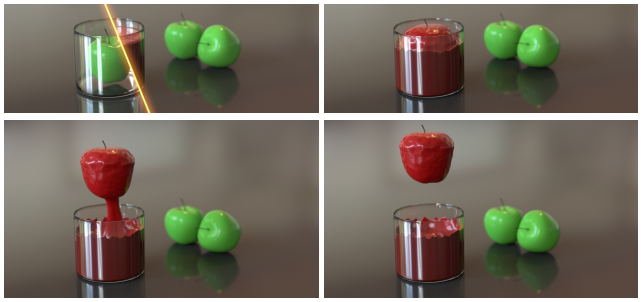

Figure 2: *An apple is pulled from liquid candy and it hardens on contact with the air, creating a candied, sticky apple.* ©*Disney.*

treatment of the Eulerian evolution of the dilational part, and careful attention to how quantities are rasterized and updated on the grid. Additionally, we couple a simple yet practical heat model to our material point solver, allowing us to drive material changes with temperature and phase.

2 Previous work

Thermodynamic variation of material properties to achieve melting and solidifying effects for visual simulation was first explored in the pioneering work of [Terzopoulos et al. 1991]. Since then, such explorations have remained very popular. A common requirement in such approaches is the unified treatment of a wide variety of material behaviors. While specialized techniques for single materials are relevant when discussing prior approaches, we primarily restrict the following literature discussion to papers that explicitly consider multiple materials with solidification and melting.

Particle-based melting. SPH is commonly used for modeling viscosity and pressure response in liquids and has been popular in graphics since [Desbrun and Gascuel 1996]. Because of its wide use, SPH has been frequently modified with more general strain computations that allow more general stress response. For example, [Solenthaler et al. 2007] simply use standard SPH interpolation to create a continuous displacement field from per-particle world space positions that can be differentiated in material-space to obtain per-particle displacement gradients. [Becker et al. 2009] show however that this displacement differentiation approach cannot accurately resolve material rotations, so they propose a shape-matching [Müller et al. 2005] approach instead. [Chang et al. 2009] handle viscoelastic and melting flow by computing the strain using a convenient Eulerian evolution (as in [Goktekin et al. 2004]) that only requires SPH interpolation of the velocities in world space. A number of other SPH methods have used Moving Least Squares to compute the strain. [Keiser et al. 2005] and [Müller et al. 2004] handle the transition from solid to fluid by including both traditional SPH-based pressure forces with elastic forces defined from an elastic potential defined via the Moving Least Squares approximation to the deformation gradient. While Moving Least Squares approaches do not suffer from the rotational artifacts encountered in the more straightforward methods of [Solenthaler et al. 2007] and [Becker et al. 2009], they are plagued by a number of failure scenarios where inversion of the associated moment matrices are not defined (e.g. co-planar and co-linear particle configurations as discussed in [Becker et al. 2009]). [Paiva et al. 2009] and [Paiva et al. 2006] avoid the need for strain computation altogether instead using non-Newtonian modifications of fluid viscosity to achieve complex fluid effects useful in melting and solidifying. Other notable uses of SPH for melting effects include [Stora et al. 1999] for lava flows, [Iwasaki et al. 2010] and [Lii and Wong 2013] for melting ice and [Lenaerts and Dutre 2009] for the treatment of porous granular materials and water.

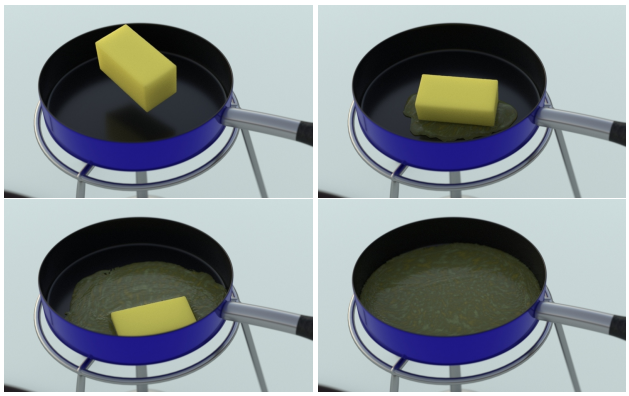

Figure 3: *Our method is able to capture many intricate features of butter melting over a hot frying pan, such as wave-like spread and micro ripples of the fluid phase, as well as effortless sliding behavior of the solid chunk on top of it.* ©*Disney.*

Mesh-based melting. Lagrangian meshes have long been popular due to trivial per-element strain computation that leads to accurate elastic behavior [Teschner et al. 2004]. However, fluid and melting behaviors necessitate topological change, requiring remeshing. [Bargteil et al. 2007] achieved efficient remeshing, [Wojtan and Turk 2008] increased efficiency and fidelity using embedded meshes, and [Wojtan et al. 2009] included the treatment of splitting. [Wicke et al. 2010] introduce a dynamic local remeshing algorithm that attempts to replace as few tetrahedra as possible, limiting the number of visual artifacts. [Clausen et al. 2013] used tetrahedron-based remeshing to melt viscoelastic solids into fluids. [Kim et al. 2006] model ice dynamics as a thin film Stefan problem and represent ice volumes with a level set method.

Grid-based melting. Eulerian methods are natural when melting into a fluid phase. However, the challenge is then the computation of elastic strain. [Goktekin et al. 2004; Losasso et al. 2006a] use an Eulerian update rule for the strain evolution. [Rasmussen et al. 2004] achieve melting effects by simply increasing viscosity in an Eulerian approach. [Wojtan et al. 2007] include erosion phase change effects with a level set representation of fluids and eroding solids. [Zhao et al. 2006] use a modified Eulerian lattice Boltzmann method to treat melting and flowing. [Wei et al. 2003] use a cellular-automata-based simplification of the physics. [Losasso et al. 2006b] couple a Lagrangian mesh representation of a solid with Eulerian representations of a fluid to treat each phase in the melting process. [Carlson et al. 2002] also combine Lagrangian and Eulerian approaches by using particles for material advection and a MAC grid for implicit viscosity and pressure projection.

Heat and phase transitions. Heat evolution is typically achieved by solving the heat equation in the world space of the system. The local temperature of the material can then be used to modify its mechanical properties. [Stora et al. 1999] varied viscosity with temperature to simulate lava flows. [Terzopoulos et al. 1991; Teschner et al. 2004; Zhao et al. 2006; Losasso et al. 2006a; Iwasaki et al. 2010; Clausen et al. 2013] model phase transition using a hard freezing temperature threshold. On the other hand, [Carlson et al. 2002; Keiser et al. 2005; Paiva et al. 2006; Solenthaler et al. 2007; Chang et al. 2009; Paiva et al. 2009; Dagenais et al. 2012] define a more smoothed material property range in the phase transition region, perhaps to model the latent heat. [Maréchal et al. 2010; Lii and Wong 2013] more correctly model phase transition including latent heat.

3 Method Overview

Goal. Our goal is to simulate a wide variety of materials, with the specific area of focus being volumetric simulation in the presence of phase change. A fully general unified simulation model is beyond the scope of our paper, and such a model would need to consider many more interactions. Researchers have considered some of these other goals with coupling [Carlson et al. 2004; Chentanez et al. 2006; Robinson-Mosher et al. 2008] and multi-material unification [Martin et al. 2010] (these citations are not exhaustive). Our focus is on heat-driven material change, in particular, because it requires handling a wide range of material behaviors and the transition within that range. We stress, however, that if a practitioner requires only one material at a time, computational efficiency might be obtained by using a specialized solver (e.g. FLIP for liquids).

MPM limitations. [Stomakhin et al. 2013] demonstrates that material point methods occupy an interesting middle ground for simulation techniques, especially elasto-plastic materials undergoing fracture. By adding plasticity to the basic constitutive model energy in [Stomakhin et al. 2012], they show that a range of compressible materials (like snow) can be simulated. While incompressibility can be approached by increasing the Poisson's ratio, at some point locking can occur [Mast et al. 2012]. At that point one might decide to use a much simpler incompressible FLIP method. However, more generally speaking, numerical systems can usually be formulated using hard constraints or soft constraints. Soft constraints can vary stiffness, but at sufficiently high stiffness, hard constraint formulations become efficient and necessary. For example, stiffer mass-spring systems can approach rigidity, but practitioners usually turn to the reduced-coordinate rigid body systems. Analogously, liquids can be simulated using equation-of-state SPH, but Incompressible SPH [Solenthaler and Pajarola 2009] is often more efficient. Regardless, in the presence of material transition, it becomes difficult to switch different parts of the domain between hard constraints and soft constraints, so soft constraint methods are used everywhere. This serves to motivate the key idea of the paper, to bring some of the efficiency of hard-constraint incompressible FLIP methods to soft-constraint MPM techniques like [Stomakhin et al. 2013].

Contributions. Our basic approach is to combine the projection ideas present in incompressible FLIP with the rich constitutive material properties of MPM to get a very flexible solver. Our particular contributions are the following:

1. We carefully model *heat in the context of MPM* by solving the heat equation on a background grid. Using the resulting temperature and phase, we can vary material properties like the Young's modulus and Poisson ratio. To solve for specifically problematic parameters that cause MPM locking, we develop a *generalized Chorin-style projection*, further requiring a *MAC-style staggered MPM formulation*.

2. We further show that a *deviatoric/dilational splitting of the constitutive model* naturally allows for this while facilitating arbitrary variation from compressible to incompressible.

3. We also show that sharp phase transitions also benefit from a deviatoric strain-based energy density function because it prevents energy gain when transitioning from fluid to solid.

We now proceed with the details of our solver. A visual diagram of our method is shown in Figure 4. In Section 4 we derive the physical equations for the mechanical evolution and heat transfer, as well as our splitting scheme. In Section 5 we discuss the details of our algorithm. Finally we present results in Section 6 and discussion in Section 7.

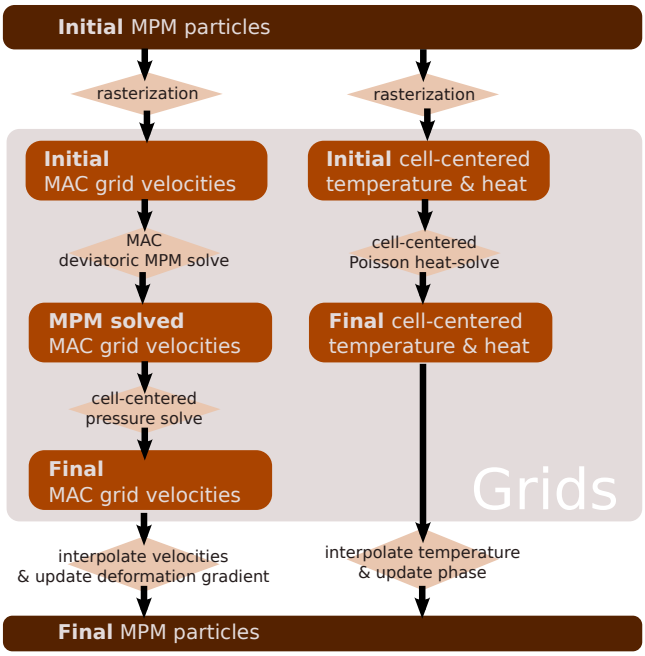

Figure 4: *Our method benefits from the interplay of grids and particles. In parallel with our mechanical evolution we have a thermodynamic evolution that also uses grids as a scratchpad.*

4 Physical Model

We describe the mapping from points in an initial material configuration $\boldsymbol{X}$ to their deformed state $\boldsymbol{x}$ by a transform $\boldsymbol{x} = \boldsymbol{\phi}(\boldsymbol{X})$. We use the notation $\boldsymbol{F} = \partial\boldsymbol{\phi}/\partial\boldsymbol{X}$ to describe the Jacobian (or deformation gradient) of the mapping. Material motion is governed by conservation of mass, conservation of momentum and the elasto-plastic constitutive relation

$$\frac{D\rho}{Dt} = 0, \quad \rho\frac{D\boldsymbol{v}}{Dt} = \nabla\cdot\boldsymbol{\sigma} + \rho\boldsymbol{g}, \quad \boldsymbol{\sigma} = \frac{1}{J}\frac{\partial\Psi}{\partial\boldsymbol{F}_E}\boldsymbol{F}_E^T, \qquad (1)$$

where ρ is density, t is time, $\boldsymbol{v}$ is velocity, $\boldsymbol{\sigma}$ is the Cauchy stress, $\boldsymbol{g}$ is the gravity, Ψ is the elasto-plastic potential energy density, $\boldsymbol{F}_E$ is the elastic part of the deformation gradient $\boldsymbol{F}$ and $J = \det(\boldsymbol{F})$ (see e.g. [Bonet and Wood 1997]).

The heat flow is given by

$$\rho\frac{Du}{Dt} = -\nabla\cdot\boldsymbol{q}, \quad \boldsymbol{q} = -\kappa\nabla T, \quad c = \frac{du}{dT}, \qquad (2)$$

where u is stored heat energy per unit mass, T is temperature, $\boldsymbol{q}$ is heat flux, κ is heat conductivity in accordance with Fourier's Law, and c is heat capacity per unit mass (see e.g. [Gonzalez and Stuart 2008]). Eliminating u and $\boldsymbol{q}$ leads to the heat equation

$$\rho c\frac{DT}{Dt} = \nabla\cdot(\kappa\nabla T). \qquad (3)$$

This is a simplified model in that we assume no transfer between the mechanical and heat energy of the system (and hence u is a function of T only). Even so, this is the most popular approach for simulating heat transfer in graphics, as can be seen from the papers listed in the previous work. Also, instead of representing volumetric heat source terms we use heat boundary conditions: Dirichlet or Neumann, depending on the desired behavior.

To complete our physical model we must form a thermo-mechanical model by bringing our heat and mechanical systems

together. This is accomplished by varying Ψ with temperature and phase. In particular, we use different expressions for Ψ depending on whether the material is in a solid or liquid state. It is worth noting that stable transition between two phases requires the careful treatment discussed later.

4.1 Heat flow and phase transition

We discretize the temperature evolution in time from (3) as

$$T^{n+1} - T^n = \frac{\Delta t}{\rho^n c^n} \nabla \cdot (\kappa^n \nabla T^{n+1}). \tag{4}$$

Note, however, that this equation describes temperature evolution only within one phase. Phase transition is a separate process in the sense that it requires extra heat, so called *latent heat*, which cannot be observed as a temperature change. Specifically, the latent heat of fusion L of an object is the heat required to transfer it from a solid to a liquid state isothermally at the freezing point of the material (see e.g. [Serway and Jewett 2009]). Thus, the transition does not happen instantly at the freezing point, and the importance of capturing this effect is discussed in [Lii and Wong 2013].

Some researchers mimic the effect of latent heat by expanding the temperature range in the vicinity of the freezing point and introducing separate temperatures for melting and freezing [Carlson et al. 2002; Keiser et al. 2005; Paiva et al. 2006; Solenthaler et al. 2007; Paiva et al. 2009; Chang et al. 2009; Dagenais et al. 2012]. While this approach is sufficient for handling phase transition of a single material, it is not generally applicable to mixtures of materials with different thermal properties, since the expanded temperature ranges would not necessarily agree. We thus will follow the approach of [Maréchal et al. 2010; Lii and Wong 2013] to accurately handle the effect latent heat in the multimaterial case. We discuss our latent heat treatment in Section 5.9.

4.2 Constitutive model

For a realistic treatment of melting and freezing, we require a suitable and well-behaved handling of plasticity and transition between liquid and solid phases of the materials. Following the multiplicative plasticity treatment of [Stomakhin et al. 2013], we separate $\boldsymbol{F}$ into an elastic part $\boldsymbol{F}_E$ and a plastic part $\boldsymbol{F}_P$ so that $\boldsymbol{F} = \boldsymbol{F}_E \boldsymbol{F}_P$. With this separation, we base our constitutive model on the elasto-plastic fixed co-rotational energy density function [Stomakhin et al. 2012; Stomakhin et al. 2013]

$$\Psi(\boldsymbol{F}_E) = \Psi_\mu(\boldsymbol{F}_E) + \Psi_\lambda(J_E), \tag{5}$$

where

$$\Psi_\mu(\boldsymbol{F}_E) = \mu \|\boldsymbol{F}_E - \boldsymbol{R}_E\|_F^2, \quad \Psi_\lambda(J_E) = \frac{\lambda}{2}(J_E - 1)^2, \tag{6}$$

$J_E = \det(\boldsymbol{F}_E)$, and $\boldsymbol{R}_E$ is the rotation from the polar decomposition of $\boldsymbol{F}_E$. This constitutive model is known to be suitable for solids, where μ and λ are typically set from Young's modulus and Poisson's ratio of the material. Furthermore, letting $\mu = 0$ makes the energy density depend only on the local volume change and thus is suitable for liquids, both compressible and incompressible (in the $\lambda \to \infty$ limit). In fact, in this case it can be shown that the Cauchy stress is a scalar pressure. Specifically, J_E measures relative volume change, and Ψ_λ penalizes it, facilitating volume preservation.

Note however, that Ψ_μ is not completely orthogonal to Ψ_λ in the sense that it also penalizes volume change. In addition, it penalizes deviatoric strains which Ψ_λ is oblivious to. Thus, simply overriding Ψ_μ in (5) when changing phase is unsuitable for freezing, as this transition would result in a sudden large increase in potential energy and produce popping artifacts. Clearly this energy increase must be avoided if freezing is to be possible.

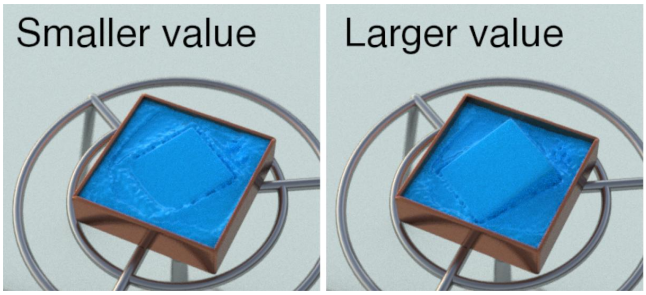

Figure 5: *Changing the value of latent heat affects the rate of phase transition, demonstrated by this melting wax example.* ©*Disney.*

In order to better understand where the energy increase comes from consider the dilational $(J_E)^{\frac{1}{d}}\boldsymbol{I}$ and deviatoric $(J_E)^{-\frac{1}{d}}\boldsymbol{F}_E$ parts of $\boldsymbol{F}_E$, where d is the dimension and $\boldsymbol{I}$ is the identity matrix. The first source of energy is the consequence of the deviatoric component of $\boldsymbol{F}_E$. The deviatoric part is not used in Ψ_λ and would generally change quite drastically with the flow. To remedy this, we note that fluids are almost perfectly plastic with respect to deviatoric strain. We incorporate this into our model by clearing the deviatoric component from $\boldsymbol{F}_E$ immediately after it is updated by letting $\boldsymbol{F}_E \leftarrow (J_E)^{\frac{1}{d}}\boldsymbol{I}$ at the end of each time step in the fluid phase.

This fluid plasticity does not completely eliminate the problem, since Ψ_μ is nonzero even if $\boldsymbol{F}_E$ contains only a dilational component. To address this, we eliminate the dilational component explicitly from Ψ_μ. This is commonly done for nearly-incompressible materials [Bonet and Wood 1997] and helps allow for arbitrarily large λ. So, we define an alternative energy density function

$$\hat{\Psi}(\boldsymbol{F}_E) = \hat{\Psi}_\mu(\boldsymbol{F}_E) + \Psi_\lambda(J_E) \tag{7}$$

$$\text{where} \quad \hat{\Psi}_\mu(\boldsymbol{F}_E) = \Psi_\mu(J_E^{-\frac{1}{d}}\boldsymbol{F}_E). \tag{8}$$

The derivatives of Ψ_μ and Ψ_λ are as in [Stomakhin et al. 2013], and the chain rule gives us the deviatoric stress $\boldsymbol{\sigma}_\mu = \frac{1}{J}\frac{\partial \hat{\Psi}_\mu}{\partial \boldsymbol{F}_E}\boldsymbol{F}_E^T$ where for clarity $\frac{\partial \hat{\Psi}_\mu}{\partial \boldsymbol{F}_E}(\boldsymbol{F}_E)$ is an evaluation of a function at $\boldsymbol{F}_E$. See the supporting technical document for details related to the derivative terms arising from the chain rule.

In general, material parameters, e.g. μ and λ, can be defined as functions of the current temperature in addition to them being functions of the current phase, however in practice we found that keeping them constant with T and letting $\mu = 0$ in the fluid phase was sufficient to produce visually compelling results.

4.3 Pressure Splitting

The model as stated would handle some material variation, but locking could occur in highly incompressible materials. This section shows how to prevent locking by transforming our solid model into a more fluid-like form, whose resulting discretization will be much more efficient. This process is analogous to fluid-only methods that are derived by starting with general continuum stresses together with simplifying assumptions that lead to a pressure equation of state. We will follow a similar strategy, albeit without the fluid-only simplifying assumption by starting with our hyperelastic stress given in (5). Although this derivation ultimately yields the commonly used pressure $p = k(\rho - \rho_0)$, where k is a stiffness, ρ is pressure and ρ_0 is the rest density, that connection must be proven.

4.4 Pressure

The problematic term for highly incompressible materials is Ψ_λ. However, we note that this term gives rise to a dilational (constant diagonal) Cauchy stress as

$$\boldsymbol{\sigma}_\lambda = \frac{1}{J}\left(\frac{\partial \Psi_\lambda}{\partial J_E}\frac{\partial J_E}{\partial \boldsymbol{F}_E}\right)\boldsymbol{F}_E^T = \frac{1}{J}\frac{\partial \Psi_\lambda}{\partial J_E} J_E \boldsymbol{F}_E^{-T}\boldsymbol{F}_E^T = -p\boldsymbol{I}, \quad (9)$$

where

$$p := -\frac{1}{J_P}\frac{\partial \Psi_\lambda}{\partial J_E} = -\frac{1}{J_P}\lambda(J_E - 1) \quad (10)$$

It is interesting to note that in the absence of plasticity $J = J_E$, $J_p = 1$, and $J = \rho/\rho_0$, making (10) reduce to $p = -\lambda(\rho/\rho_0 - 1)$, the traditional SPH equation of state [Monaghan 1992].

4.5 Temporal evolution

Even though p is related to our other variables, by treating it as an unknown, we can achieve a splitting, analogous to Chorin-style projection. The main difference is that we are not restricted to fully incompressible materials, and we instead handle the full spectrum. To derive the splitting consider the time evolution of pressure

$$\frac{Dp}{Dt} = -\frac{1}{J_P}\frac{\partial^2 \Psi_\lambda}{\partial J_E^2}\frac{DJ_E}{Dt}. \quad (11)$$

Since $J = J_E J_P$ and $\frac{DJ}{Dt} = J\nabla\cdot\boldsymbol{v}$ (see e.g. [Gonzalez and Stuart 2008]), we have $\frac{DJ_E}{Dt} = J_E\nabla\cdot\boldsymbol{v}$ and therefore

$$\frac{Dp}{Dt} = -\frac{1}{J_P}\frac{\partial^2 \Psi_\lambda}{\partial J_E^2} J_E \nabla\cdot\boldsymbol{v} = -\frac{\lambda J_E}{J_P}\nabla\cdot\boldsymbol{v}. \quad (12)$$

4.6 Discretization

In addition, with the definition of p from (9), our force balance equation takes the fluid-like form

$$\rho\frac{D\boldsymbol{v}}{Dt} = \nabla\cdot\boldsymbol{\sigma} + \rho\boldsymbol{g} = \nabla\cdot\boldsymbol{\sigma}_\mu - \nabla p + \rho\boldsymbol{g}, \quad (13)$$

where $\boldsymbol{\sigma}_\mu$ is the component of stress from Ψ_μ. We discretize the system of equations (12) and (13) as

$$\frac{p^{n+1} - p^n}{\Delta t} = -\frac{\lambda^n J_E^n}{J_P^n}\nabla\cdot\boldsymbol{v}^{n+1}, \quad (14)$$

$$\frac{\boldsymbol{v}^{n+1} - \boldsymbol{v}^n}{\Delta t} = \frac{1}{\rho^n}\nabla\cdot\boldsymbol{\sigma}_\mu - \frac{1}{\rho^n}\nabla p^{n+1} + \boldsymbol{g}. \quad (15)$$

Note that we can replace the material derivative with a simple finite difference in time because advection will be done in a Lagrangian manner using MPM.

In order to solve the system (14) and (15) we split the pressure application in (15) from the other forces by introducing an intermediate $\boldsymbol{v}^\star$

$$\frac{\boldsymbol{v}^\star - \boldsymbol{v}^n}{\Delta t} = \frac{1}{\rho^n}\nabla\cdot\boldsymbol{\sigma}_\mu + \boldsymbol{g}, \quad (16)$$

$$\frac{\boldsymbol{v}^{n+1} - \boldsymbol{v}^\star}{\Delta t} = -\frac{1}{\rho^n}\nabla p^{n+1}. \quad (17)$$

Taking the divergence of (17) and eliminating $\nabla\cdot\boldsymbol{v}^{n+1}$ using (14) yields

$$\frac{J_P^n}{\lambda^n J_E^n}\frac{p^{n+1}}{\Delta t} - \Delta t\nabla\cdot\left(\frac{1}{\rho^n}\nabla p^{n+1}\right) = \frac{J_P^n}{\lambda^n J_E^n}\frac{p^n}{\Delta t} - \nabla\cdot\boldsymbol{v}^\star. \quad (18)$$

We use $p^n = -\frac{1}{J_P^n}\lambda^n(J_E^n - 1)$ for the right hand side.

Note that the discrete system for the pressure will be symmetric positive definite and similar to a discrete heat equation for moderate λ. As λ is increased to the incompressible limit, the pressure equation is then the standard Poisson equation seen in Chorin-style projections [Chorin 1968]. This is similar in spirit to the implicit treatment of the compressible Euler equations in [Kwatra et al. 2009]. While the introduction of an auxiliary pressure unknown is common in incompressible elasticity (see e.g. [Bonet and Wood 1997]), it would generally be coupled with the velocity unknowns (see e.g. [Mast et al. 2012]). Our introduction of the implicit treatment based on the evolution of pressure (11) is novel and drastically improves the efficiency of the approach because it decouples the pressure from the nonlinear equations for velocity unknowns.

5 Algorithm

Here we describe the discretization details in our algorithm. We outline each step required to advance one time step in the simulation (see Figure 4 for a schematic overview). We can think of this process as updating the state (itemized in Table 1) from time t^n to time t^{n+1}. The process uses a background MAC grid and combines standard aspects of traditional MPM and FLIP solvers. Specifically, after the particle state is transferred to the grid, the deviatoric forces are first discretized with implicit MPM in accordance with (16). This step results in an intermediate velocity field whose divergence is used in the right hand side of the implicit equation for the dilational part in (18). The dilational part is treated with the generalized Chorin-style projection over the MAC grid and the intermediate velocity is then given a pressure correction in accordance with (17). The inclusion of the heat transfer effects only requires an additional heat equation solve per time step. We discuss the specific details of each step in the algorithm in the following subsections, which can be summarized as:

1. Apply plasticity from previous timestep (Section 5.1)
2. Compute interpolation weights (Section 5.2)
3. Rasterize particle data to grid (Section 5.3)
4. Classify cells (Section 5.4)
5. Compute MPM forces (Section 5.5.1)
6. Process grid collisions (Section 5.6)
7. Apply implicit MPM update (Section 5.5.2)
8. Project velocities (Section 5.7)
9. Solve heat equation (Section 5.8)
10. Update particle state from grid (Section 5.9)
11. Process particle collisions and update particle positions (Section 5.10)

5.1 Apply plasticity from previous timestep

For simplicity, it is common for graphics researchers to apply a heuristic plastic-yield criterion for compressible elastic materials, because there is considerable leeway in visual applications [Irving et al. 2004; Stomakhin et al. 2013]. However, in the case of nearly incompressible materials, the plastic flow should also be nearly incompressible. We therefore provide a simple procedure for guaranteeing $J_P \equiv \det(\boldsymbol{F}_P) = 1$ for nearly incompressible materials. We note that more accurate plasticity models from the engineering literature (such as von Mises yield criteria) also have the property that $J_P = 1$ as a consequence of rate independence (see [Bonet and Wood 1997; Goktekin et al. 2004; Bargteil et al. 2007]). We begin by adjusting $\boldsymbol{F}_E$ and $\boldsymbol{F}_P$ so that the singular values of $\boldsymbol{F}_E$ are restricted to the interval $[1-\theta_c, 1+\theta_s]$ as in [Stomakhin et al. 2013]. We then apply the correction $\boldsymbol{F}_E \leftarrow (J_P)^{1/d}\boldsymbol{F}_E$ and

Notation	Description	Is Constant
$\boldsymbol{x}_p$	Position	Not constant
$\boldsymbol{v}_p$	Velocity	Not constant
m_p	Mass	Constant
V_p^0	Initial volume	Constant
$\boldsymbol{F}_{Ep}$	Elastic part of $\boldsymbol{F}_p$	Not constant
$\boldsymbol{F}_{Pp}$	Plastic part of $\boldsymbol{F}_p$	Not constant
μ_p	Lamé parameter μ	Depends on T_p and phase
λ_p	Lamé parameter λ	Depends on T_p, but not phase
T_p	Temperature	Not constant
U_p	Transition heat	Not constant (Sec. 5.9)
c_p	Heat capacity per unit mass	Depends on T_p and phase
κ_p	Heat conductivity	Depends on T_p and phase
L_p	Latent heat	Constant
ζ_p	Phase	Depends on T_p and U_p (Sec.5.9)

Table 1: *Quantities stored on each particle.*

$\boldsymbol{F}_P \leftarrow (J_P)^{-1/d}\boldsymbol{F}_P$, which ensures that $\boldsymbol{F}_P$ is purely deviatoric, or equivalently, $J_P = 1$.

5.2 Compute interpolation weights

In order to transfer data from particles to MAC faces and MAC cell centers, we need multiple sets of interpolation weights per particle. Basically, we have d face-centered grids, one for each dimension, and one cell-centered grid. The procedure of computing the weights is identical for all of these grids and follows [Steffen et al. 2008]. The grids however are offset with respect to each other, which leads to different weight values for each grid. Below, we introduce a common notation for all offset grids and describe a way to procedurally calculate the weights.

We express the fact that the $d+1$ grids are offset with respect to each other by considering their base point (x_{0a}, y_{0a}, z_{0a}) (lower-left point in 2D), where $a \in \{x, y, z\}$ indicates velocity components for each of the face grids and $a = \star$ represents the pressure grid. Up to some translation vector, we have $(x_{0x}, y_{0x}, z_{0x}) = (-\frac{h}{2}, 0, 0)$, $(x_{0y}, y_{0y}, z_{0y}) = (0, -\frac{h}{2}, 0)$, $(x_{0z}, y_{0z}, z_{0z}) = (0, 0, -\frac{h}{2})$, and $(x_{0\star}, y_{0\star}, z_{0\star}) = (0, 0, 0)$, where h is the grid spacing. See Figure 6 (left) for an illustration of the MAC grids. Now, given a grid of spacing h with cell indices $\mathbf{c} = (i, j, k)$ with points located at $\boldsymbol{x}_{\mathbf{c}a} = (x_i, y_j, z_k) = (x_{0a}+ih, y_{0a}+jh, z_{0a}+kh)$ we can define interpolation of an arbitrary particle position $\boldsymbol{x}_p = (x_p, y_p, z_p)$. As in [Steffen et al. 2008], we define a multidimensional separable kernel from the one-dimensional cubic B-spline

$$N(x) = \begin{cases} \frac{1}{2}|x|^3 - x^2 + \frac{2}{3}, & 0 \le |x| < 1 \\ -\frac{1}{6}|x|^3 + x^2 - 2|x| + \frac{4}{3}, & 1 \le |x| < 2 \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

as $N^h_{\mathbf{c}a}(\boldsymbol{x}_p) = N(\frac{1}{h}(x_p - x_{ia}))N(\frac{1}{h}(y_p - y_{ja}))N(\frac{1}{h}(z_p - z_{ka}))$.

For a more compact notation later on we will use $\mathbf{i}$ as an index into MAC grid faces, and use $\mathbf{c}$ for indexing cell-centered quantities. E.g. $v_\mathbf{i}$ stands for the velocity field component at face $\mathbf{i}$, and $p_\mathbf{c}$ is the pressure value at the center of cell $\mathbf{c}$. With this the interpolation weight of particle $\boldsymbol{x}_p$ is $w_{\mathbf{i}p} = N^h_{\mathbf{c}(\mathbf{i})a(\mathbf{i})}(\boldsymbol{x}_p)$ with respect to face $\mathbf{i}$ and $w_{\mathbf{c}p} = N^h_{\mathbf{c}\star}(\boldsymbol{x}_p)$ with respect to cell $\mathbf{c}$. Here $a(\mathbf{i})$ and $\mathbf{c}(\mathbf{i})$ are the dimension component and cell index associated with face $\mathbf{i}$ respectively. Alternatively face index can be uniquely identified by a cell and an axis as $\mathbf{i} = \mathbf{i}(\mathbf{c}, a)$, for $a \in \{x, y, z\}$. Similarly, we define $\nabla w_{\mathbf{i}p} = \nabla N^h_{\mathbf{c}(\mathbf{i})a(\mathbf{i})}(\boldsymbol{x}_p)$ and $\nabla w_{\mathbf{c}p} = \nabla N^h_{\mathbf{c}\star}(\boldsymbol{x}_p)$. The various components and their associated values are summarized in the following table:

Grid	a	Base	Weight
cell	$\star$	$(0, 0, 0)$	$w_{\mathbf{c}p} = N^h_{\mathbf{c}\star}(\boldsymbol{x}_p)$
x-offset	x	$(-\frac{h}{2}, 0, 0)$	$w_{\mathbf{i}(\mathbf{c},x)p} = N^h_{\mathbf{c}x}(\boldsymbol{x}_p)$
y-offset	y	$(0, -\frac{h}{2}, 0)$	$w_{\mathbf{i}(\mathbf{c},y)p} = N^h_{\mathbf{c}y}(\boldsymbol{x}_p)$
z-offset	z	$(0, 0, -\frac{h}{2})$	$w_{\mathbf{i}(\mathbf{c},z)p} = N^h_{\mathbf{c}z}(\boldsymbol{x}_p)$

5.3 Rasterize particle data to grid

We rasterize data to the grid using the interpolation weights from Section 5.2. Mass is first rasterized to the grid faces as

$$m^n_\mathbf{i} = \sum_p w^n_{\mathbf{i}p} m_p.$$

These face densities allow us to normalize the interpolation of velocity and thermal conductivity as

$$A^n_\mathbf{i} = \sum_p w^n_{\mathbf{i}p} m_p A^n_p \quad \text{for} \quad A \in \{v, \kappa\}.$$

We repeat the process at cell centers, computing cell masses $m^n_\mathbf{c} = \sum_p w^n_{\mathbf{c}p} m_p$ followed by

$$B^n_\mathbf{c} = \frac{1}{m^n_\mathbf{c}} \sum_p w^n_{\mathbf{c}p} m_p A^n_p \quad \text{for} \quad B \in \{J, J_E, c, T, \lambda^{-1}\}$$

noting that rasterizing λ^{-1} rather than λ is important for stability. [1] Finally, we set $J^n_{P\mathbf{c}} = \frac{J^n_\mathbf{c}}{J^n_{E\mathbf{c}}}$.

5.4 Classify cells

We represent our collision objects as level sets and assign each collision object a temperature. We begin the collision processing by checking all faces for collisions. A MAC face is colliding if the level set computed by any collision object is negative at the face center. If it is colliding, we flag the face as colliding. For convenience and consistency in other parts of the algorithm, we classify each MAC cell as *empty*, *colliding*, or *interior*. A cell is marked as colliding if all of its surrounding faces are colliding. Otherwise, a cell is interior if the cell and all of its surrounding faces have mass. All remaining cells are empty. See Figure 6 (left). Colliding cells are either assigned the temperature of the object it collides with or a user-defined spatially-varying value depending on the setup. If the free surface is being enforced as a Dirichlet temperature condition, the ambient air temperature is recorded for empty cells. No other cells require temperatures to be recorded at this stage.

5.5 MPM velocity update

In our deviatoric/dilational splitting of the material response, the deviatoric forces are discretized with implicit MPM, and the dilation part is discretized with the generalized Chorin-style projection. Using the common notation from a projection method, we can think of the the implicit MPM step as updating rasterized grid-based velocities $v^n_\mathbf{i}$ to $v^\star_\mathbf{i}$ in accordance with (16). The last step for grid velocities is to apply the pressure correction, computed using (18), to $v^\star_\mathbf{i}$ to obtain $v^{n+1}_\mathbf{i}$ in accordance with (17). In this section and the following subsections we outline the procedure for computing $v^\star_\mathbf{i}$. The first step is to compute the MPM force.

[1] The relationship between J_E and λ results in a balance in the pressure $-\frac{\lambda}{J_P}(J_E - 1)$. Unfortunately, averaging J_E and λ through rasterization might destroy this balance, creating an artificially large pressure. Estimating λ with a harmonic average, or equivalently, rasterizing λ^{-1} and computing $\lambda_\mathbf{c} = 1/\lambda^{-1}_\mathbf{c}$, resolves this problem.

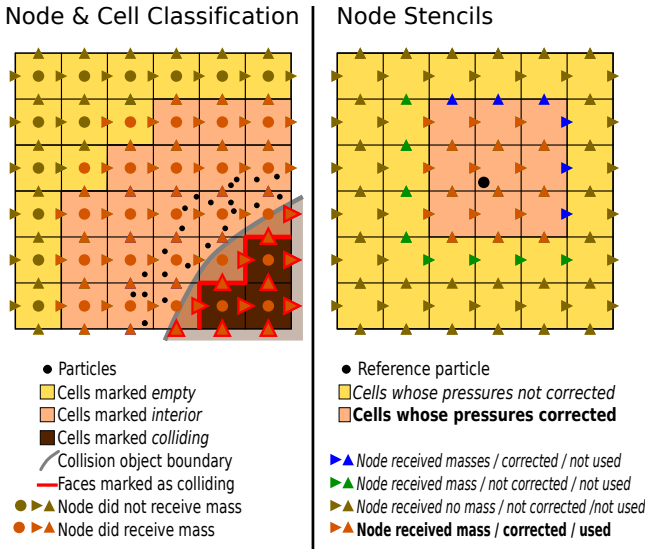

Figure 6: ***Left figure*** *illustrates cell classification criteria. Note that faces marked "colliding" are Neumann faces for the Poisson solve and yellow cells marked "colliding" are Dirichlet cells for the Poisson solve.* ***Right figure*** *shows stencils for a single reference particle. The particle contributes to the green, blue, and orange faces, the pressure solve only corrects orange and blue faces, but our quadratic interpolation touches only the orange faces.*

Following [Stomakhin et al. 2013], we discretize the deviatoric forces via a potential energy. This naturally facilitates an implicit treatment with symmetric linearization. We denote the location of grid face $\mathbf{i}$ as $\boldsymbol{x}_\mathbf{i}$. If we interpret our Eulerian MAC grid as though it were Lagrangian, we would estimate that after Δt, this face would have moved to $\hat{\boldsymbol{x}}_\mathbf{i} = \boldsymbol{x}_\mathbf{i} + \Delta t v_\mathbf{i}^\star \boldsymbol{e}_{a(\mathbf{i})}$, where $\boldsymbol{e}_{a(\mathbf{i})}$ is the basis vector in the direction corresponding to the MAC velocity component $v_\mathbf{i}^\star$. If we denote the vector of all $\hat{\boldsymbol{x}}_\mathbf{i}$ as $\hat{\boldsymbol{x}}$, then we can think of it as depending on the vector of all face velocities $v_\mathbf{i}^\star$ which we can denote as $\boldsymbol{v}^\star$. Or, $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}(\boldsymbol{v}^\star)$. Note that this interpretation is for convenience in computing forces and force derivatives as we do not actually move our grid. Since we only really have one degree of freedom in $\hat{\boldsymbol{x}}_\mathbf{i}$, we will denote it as $\hat{x}_\mathbf{i} = (\hat{\boldsymbol{x}}_\mathbf{i})_{a(\mathbf{i})}$ and $\hat{x}_\mathbf{i} = \hat{x}_\mathbf{i}(v_\mathbf{i}^\star) = (\hat{\boldsymbol{x}}_\mathbf{i})_{a(\mathbf{i})} + \Delta t v_\mathbf{i}^\star$.

The deviatoric potential energy is

$$\Phi_\mu(\hat{\boldsymbol{x}}) = \sum_p V_p^0 \hat{\Psi}_\mu(\hat{\boldsymbol{F}}_{Ep}(\hat{\boldsymbol{x}})), \tag{20}$$

where V_p^0 is the initial volume occupied by particle p and $\hat{\boldsymbol{F}}_{Ep}$ is the elastic part of the deformation gradient of particle p. $\hat{\boldsymbol{F}}_{Ep}$ depends on $\hat{\boldsymbol{x}}$ as in [Sulsky et al. 1995]

$$\hat{\boldsymbol{F}}_{Ep}(\hat{\boldsymbol{x}}) = \left(\boldsymbol{I} + \sum_\mathbf{i} (\hat{\boldsymbol{x}}_\mathbf{i} - \boldsymbol{x}_\mathbf{i})(\nabla w_{\mathbf{i}p}^n)^T\right) \boldsymbol{F}_{Ep}^n. \tag{21}$$

5.5.1 MPM forces

The force component $f_\mathbf{i}$ at face $\mathbf{i}$ is given by $f_\mathbf{i} = -\frac{\partial \Phi}{\partial \hat{x}_\mathbf{i}} = -\frac{\partial \Phi}{\partial \hat{\boldsymbol{F}}_{Ep}} \frac{\partial \hat{\boldsymbol{F}}_{Ep}}{\partial \hat{\boldsymbol{x}}_\mathbf{i}} \frac{\partial \boldsymbol{x}_\mathbf{i}}{\partial x_\mathbf{i}}$, or

$$f_\mathbf{i}(\hat{\boldsymbol{x}}) = -\sum_p V_p^0 \boldsymbol{e}_{a(\mathbf{i})}^T \frac{\partial \Psi}{\partial \boldsymbol{F}_E}(\hat{\boldsymbol{F}}_{Ep}(\hat{\boldsymbol{x}}))(\boldsymbol{F}_{Ep}^n)^T \nabla w_{\mathbf{i}p}^n. \tag{22}$$

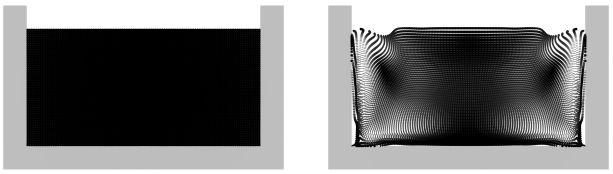

Figure 7: *Simulation of a stationary pool with (left) and without (right) density correction. Without correction the faces near collision objects appear lighter which causes them to rise under gravity.*

With these forces, we compute the right hand side for our MPM treatment

$$b_\mathbf{i} = v_\mathbf{i}^n + \frac{\Delta t}{m_\mathbf{i}} f_\mathbf{i} + \Delta t g_\mathbf{i} \sum_p w_{\mathbf{i}p}^n, \tag{23}$$

where $g_\mathbf{i}$ is the gravity component at face $\mathbf{i}$ and $f_\mathbf{i}^n = -\frac{\partial \Phi_\mu}{\partial \hat{x}_\mathbf{i}}(\hat{\boldsymbol{x}}(\mathbf{0}))$, again using the convention that $\hat{\boldsymbol{x}} = \hat{\boldsymbol{x}}(\boldsymbol{v}^\star)$.

5.5.2 Semi-implicit MPM update

We use one step of Newton's method to solve the implicit system for deviatoric and inertial force balance. This yields a (mass) symmetric system for $\boldsymbol{v}^\star$

$$\sum_\mathbf{j} \underbrace{\left(\delta_{\mathbf{ij}} + \frac{\Delta t^2}{2 m_\mathbf{i}^n} \frac{\partial^2 \Phi_\mu}{\partial \hat{x}_\mathbf{i} \partial \hat{x}_\mathbf{j}}(\hat{\boldsymbol{x}}(\mathbf{0}))\right)}_{q_{\mathbf{ij}}} v_\mathbf{j}^\star = b_\mathbf{i}, \tag{24}$$

where $q_{\mathbf{ij}}$ are the matrix $\boldsymbol{Q}$'s entries. The system is symmetric but potentially indefinite so we use the iterative conjugate residual method [Choi 2006]. This Krylov method only requires the action of $\boldsymbol{Q}$ on an arbitrary increment $\delta \boldsymbol{u}$ (comprised of scalar MAC face increments $\delta u_\mathbf{j}$). The non-trivial term is from the Hessian and can be expressed as

$$-\delta f_\mathbf{i} = \sum_\mathbf{j} \frac{\partial^2 \Phi_\mu}{\partial \hat{x}_\mathbf{i} \partial \hat{x}_\mathbf{j}}(\hat{\boldsymbol{x}}(\mathbf{0}))\delta u_\mathbf{j} = \sum_p V_p^0 \boldsymbol{e}_{a(\mathbf{i})}^T \boldsymbol{A}_p (\boldsymbol{F}_{Ep}^n)^T \nabla w_{\mathbf{i}p}^n, \tag{25}$$

where

$$\boldsymbol{A}_p = \frac{\partial^2 \Psi_\mu}{\partial \boldsymbol{F}_E^2}(\boldsymbol{F}_E(\hat{\boldsymbol{x}})) : \left(\sum_\mathbf{j} \delta u_\mathbf{j} \boldsymbol{e}_{a(\mathbf{i})} (\nabla w_{\mathbf{j}p}^n)^T \boldsymbol{F}_{Ep}^n\right). \tag{26}$$

5.6 Process grid collisions

Each face marked as colliding during the cell classification step must have its velocity corrected for collisions. We perform sticking collisions for all of our collisions, so we simply assign the velocity component from the collision body to the corresponding face on the MAC grid.

5.7 Project velocities

We discretize (18) for the pressure then use it to correct the intermediate velocities $\boldsymbol{v}^\star$. This is a discrete parabolic equation that of course reduces to a Poisson equation in the incompressible limit of $\lambda \to \infty$. In either case our discretization reduces to a symmetric positive definite system of equations. We discretize in space using the central-difference stencils naturally defined over the MAC grid. The right hand side of our system has entries $s_\mathbf{c}$ stored at MAC cell centers. We compute these as

Figure 8: *Setting a Dirichlet temperature boundary condition on the air cells allows us to melt objects from the outside.* ©*Disney.*

$s_{\mathbf{c}} = -\frac{J_E^n - 1}{\Delta t J_{E\mathbf{c}}^n} - \sum_{\mathbf{i}}^n G_{\mathbf{ic}} v_{\mathbf{i}}^\star$, where $G_{\mathbf{ic}}$ are the coefficients of the central-differenced gradient stencil. Our corresponding matrix takes the increments $\delta p_{\mathbf{c}}$ and produces the results $\delta r_{\mathbf{c}}$, where $\delta r_{\mathbf{c}} = \frac{\delta p_{\mathbf{c}} J_{P\mathbf{c}}^n}{J_{E\mathbf{c}}^n \lambda_{\mathbf{c}}^n \Delta t} + \Delta t \sum_{\mathbf{i}} \sum_{\mathbf{c}'} \frac{1}{\rho_{\mathbf{i}}^n} G_{\mathbf{ic}} G_{\mathbf{ic}'} \delta p_{\mathbf{c}'}$ and $\rho_{\mathbf{i}}^n = \frac{m_{\mathbf{i}}^n}{V_{\mathbf{i}}^n}$ is the mass density at face $\mathbf{i}$. $m_{\mathbf{i}}^n$ is the mass at the face and $V_{\mathbf{i}}^n$ is a control volume around the face whose formula we describe below. Once we have solved for the pressure, we apply the pressure correction to the velocities using $v_{\mathbf{i}}^{n+1} = v_{\mathbf{i}}^\star - \Delta t \sum_{\mathbf{c}} \frac{1}{\rho_{\mathbf{i}}^n} G_{\mathbf{ic}} p_{\mathbf{c}}$.

The discretization of $G_{\mathbf{ic}}$ corresponds to a simple voxelized, central-differenced gradient operator. We enforce homogeneous Dirichlet pressure boundary conditions at cells that have been marked as empty and homogeneous Neumann boundary conditions at faces adjacent to cells that have been marked as colliding.

Degrees of freedom near collision objects do not have as many neighboring particles as interior degrees of freedom, since part of their influence is covered by a collision object. This causes these faces to appear lighter, which would in turn cause them to rise under gravity without careful definition of $\rho_{\mathbf{i}}^n$. We prevent such phenomena by computing control volumes that accurately represent the portion of the domain associated with a face. This is done as $V_{\mathbf{i}}^n = \sum_{\mathbf{c}} \int_{\Omega_{\mathbf{c}}} \chi_{\mathbf{c}} N_{\mathbf{c(i)}a(\mathbf{i})}^h(\boldsymbol{x}) d\boldsymbol{x}$ where $\Omega_{\mathbf{c}}$ is the interior of MAC cell $\mathbf{c}$ and $\chi_{\mathbf{c}} = 1$ if cell $\mathbf{c}$ is marked as interior and $\chi_{\mathbf{c}} = 0$ otherwise. This is an approximation to $\int_{\Omega^n} N_{\mathbf{i}}^h d\boldsymbol{x}$ where Ω^n is the domain encompassed by the material. This control volume is essential for accurately approximating the density near collision objects. Note that the integral described in the formula for $V_{\mathbf{i}}^n$ has only a finite number of cases, which the product structure of $N_{\mathbf{c(i)}a(\mathbf{i})}^h(\boldsymbol{x})$ makes relatively easy to tabulate. We demonstrate the effect of density correction in Figure 7.

5.8 Solve heat equation

We perform a stabilized Poisson solve to update the temperature in accordance with (4). We begin by setting the right hand side to $T_{\mathbf{c}}^n$, which is a cell-centered rasterized temperature. Our corresponding matrix takes the increments $\delta T_{\mathbf{c}}$ and produces the result $\delta T_{\mathbf{c}} + \Delta t \sum_{\mathbf{i}}^n \sum_{\mathbf{c}'} \frac{\Delta x^d}{m_{\mathbf{c}}^n c_{\mathbf{c}}^n} \kappa_{\mathbf{i}}^n G_{\mathbf{ic}} G_{\mathbf{ic}'} \delta T_{\mathbf{c}'}$. The discretization of $G_{\mathbf{ic}}$ corresponds to a simple voxelized, central-differenced gradient operator. We enforce Dirichlet temperature boundary conditions at cells that are in contact with fixed temperature bodies (like a heated pan or air) and homogeneous Neumann boundary conditions at faces adjacent to cells that can be considered empty or corresponding to insulated objects.

5.9 Update particle state from grid

Some outermost faces involved in the MPM step do not receive a correction from the projection step, and as a result they tend to have outdated velocity values (see Figure 6). To prevent errors from

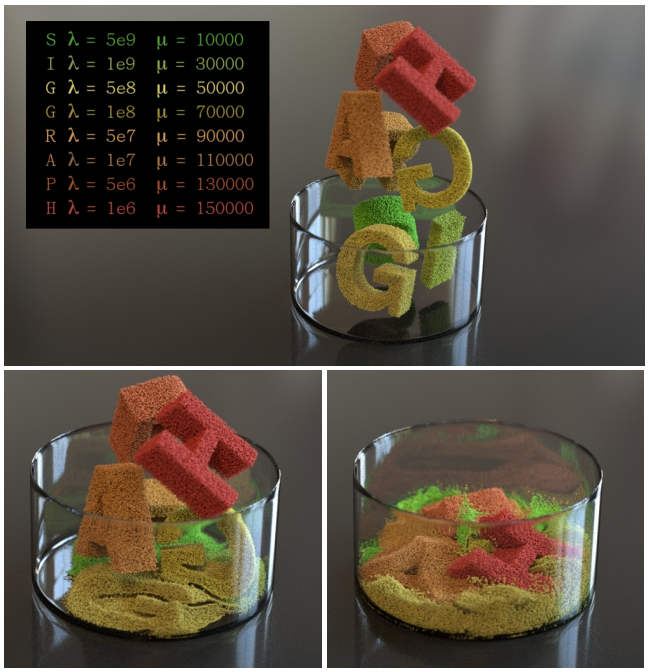

Figure 9: *Our method handles mixtures of materials with drastically different properties, ranging from compressible to (almost) incompressible. Here each letter has λ varying from 10^6 to 5×10^9, as well as varying μ and plasticity parameters.* ©*Disney.*

uncorrected velocities when transferring information back to the particles, we use a tighter quadratic stencil given by the following spline:

$$N(x) = \begin{cases} -x^2 + \frac{3}{4}, & 0 \le |x| < \frac{1}{2} \\ \frac{1}{2}x^2 - \frac{3}{2}x + \frac{9}{8}, & 1 \le |x| < \frac{3}{2} \\ 0, & \text{otherwise} \end{cases}. \quad (27)$$

We interpolate velocities back to particles using FLIP, where the PIC component is computed as $\boldsymbol{v}_p^{PIC} = \sum_{\mathbf{i}} v_{\mathbf{i}}^{n+1} w_{\mathbf{i}p}^n \boldsymbol{e}_{a(\mathbf{i})}$ and the FLIP component as $\boldsymbol{v}_p^{FLIP} = \boldsymbol{v}_p^n + \sum_{\mathbf{i}} (v_{\mathbf{i}}^{n+1} - v_{\mathbf{i}}^n) w_{\mathbf{i}p}^n \boldsymbol{e}_{a(\mathbf{i})}$. With these, the new velocities are $\boldsymbol{v}_p^{n+1} = \alpha \boldsymbol{v}_p^{FLIP} + (1-\alpha)\boldsymbol{v}_p^{PIC}$, where α is the FLIP fraction. We used $\alpha = 0.95$ in our examples.

The next step is to update $\boldsymbol{F}_{Ep}$. To do this, we must compute a velocity gradient, which we do with $\nabla \boldsymbol{v}_p^{n+1} = \sum_{\mathbf{i}} v_{\mathbf{i}}^{n+1} \boldsymbol{e}_{a(\mathbf{i})} \nabla w_{\mathbf{i}p}^T$. Normally, one would finish with the update rule $\boldsymbol{F}_{Ep}^{n+1} = (\boldsymbol{I} + \Delta t \nabla \boldsymbol{v}_p^{n+1}) \boldsymbol{F}_{Ep}^n$. We found that this occasionally leads to $J_{Ep}^{n+1} \le 0$ if the time step is too large, so we opt for a compromise between this simple rule and the ideal but expensive exponential computation $\boldsymbol{F}_{Ep}^{n+1} = e^{\Delta t \nabla \boldsymbol{v}_p^{n+1}} \boldsymbol{F}_{Ep}^n$. Instead, we use $\boldsymbol{F}_{Ep}^{n+1} = \boldsymbol{R}(\Delta t \nabla \boldsymbol{v}_p^{n+1}) \boldsymbol{F}_{Ep}^n$, where $\boldsymbol{R}(\boldsymbol{M}) = \boldsymbol{I} + \boldsymbol{M}$ if $\det(\boldsymbol{I} + \boldsymbol{M}) > 0$ and $\boldsymbol{R}(\boldsymbol{M}) = \boldsymbol{R}(\frac{1}{2}\boldsymbol{M})^2$ otherwise. Note that this is effectively a truncated geometric series of the exponential function, where we invest just enough time to keep the determinant positive. In practice, this function recurses very rarely, and the update is more robust but nearly as efficient as before. If p is a fluid particle, we finish off the update of $\boldsymbol{F}_{Ep}^{n+1}$ by removing its deviatoric component using $\boldsymbol{F}_{Ep}^{n+1} \leftarrow (J_{Ep}^{n+1})^{1/d} \boldsymbol{I}$.

Similarly, temperature gets transferred from the grid cell centers to particles as $T_p^{n+1} = \beta T_P^{FLIP} + (1-\beta) T_P^{PIC}$, where $T_p^{FLIP} = T_p^n + \sum_{\mathbf{c}} (T_{\mathbf{c}}^{n+1} - T_{\mathbf{c}}^n) w_{\mathbf{c}p}$, $T_p^{PIC} = \sum_{\mathbf{c}} T_{\mathbf{c}}^n w_{\mathbf{c}p}$ and β is the

FLIP ratio (we used $\beta = 0.95$ for our examples). As mentioned before, the heat equation and, thus, the grid-based heat update are valid only within one material phase, so cases where the temperature crosses the freezing point require special treatment. Namely, a portion of the heat the particle gets (or loses) should be spent on the phase change. To account for this effect we have an energy buffer associated with each particle of size L_p, and the particle stores the amount of heat U_p contained in that buffer, which can vary from 0 to L_p. Initially, we allow each particle to freely change its temperature according to the heat equation. But whenever the freezing point is reached, any additional temperature change is multiplied by $c_p m_p$ and added to the buffer, with the particle temperature kept unchanged. This can also be viewed as a post correction of the temperature for a particle that "illegally" crossed the freezing point. Once the buffer is completely full (particle heat $U_p = L_p$), we switch the particle phase to fluid. Conversely, if the buffer becomes empty (particle heat $U_p = 0$), we switch the particle phase to solid. Note that the phase change happens *only* when the buffer is completely full or empty, otherwise the material retains its phase from the previous timestep. This sort of hysteresis facilitates more stable phase transition, as opposed to using a hard threshold on U_p.

5.10 Process particle collisions and positions

We complete our time integration by enforcing collisions on our particles. Since we did sticking collisions with the grid, we do sticking collisions on particles as well. A particle is registered as colliding if a collision body registers a negative level set value at the location of the particle. If this occurs, the particle's velocity is set to the velocity of the collision body at that location. Finally, we update particle positions as $\boldsymbol{x}_p^{n+1} = \boldsymbol{x}_p^n + \Delta t \boldsymbol{v}_p^{n+1}$.

6 Results and examples

We have generated a number of visually interesting results using our method. Our novel splitting and rasterization techniques facilitate handling mixtures with extreme variations of material parameters. This can be seen in Figure 9 where we drop elasto-plastic SIGGRAPH letters with material properties ranging from compressible to almost incompressible with varying stiffness and plasticity parameters.

Further, our simplified yet practical heat model allows us to achieve compelling phase transition effects. Figure 10 shows hot liquid chocolate pouring on a cold solid chocolate bunny. During the process some solid melts and some liquid freezes producing intricate shapes. Figures 8 and 2 demonstrate how we can use external surface heat sources and sinks (like hot/cold air and cold frying pan) to melt and freeze different objects. Our careful treatment of the physics of phase transition using latent heat allows us to maintain sharp, yet stable, interfaces between solid and fluid phases, as can be seen in the butter melting example in Figure 3. By varying materials' thermal parameters such as heat conductivity, heat capacity and latent heat, we can control the heat flow and thus (indirectly) affect the dynamics of melting and freezing, as shown in Figure 5. To create believable lava flow solidifying into pāhoehoe shown in Figure 1, we varied the temperature of the mountain based on the distance to the lava source (the heat exchange with the air was not simulated). This way the lava would freeze more gradually, forming attractive layered shapes. We also added some variation to the particles' freezing temperature to give it a more amorphous look.

The simulation times for each of the examples are shown in Table 2. For each of those the timestep size was $\Delta t \simeq 3 \times 10^{-4}\, s$. To achieve convergence, the conjugate residual solvers for MPM, projection and heat diffusion steps normally would take under 10, 300 and 50 iterations, respectively.

Example	Particles	Grid	min/frame
SIGGRAPH letters	1.0×10^6	$96 \times 144 \times 96$	18.5
Bunny and hot stream	1.2×10^6	$170 \times 170 \times 170$	8.4
Bunny and hot air	1.2×10^6	$160 \times 160 \times 160$	11.4
Apple dip	0.8×10^6	$64 \times 128 \times 64$	11.0
Melting butter	4.2×10^6	$128 \times 128 \times 128$	14.5
Lava	3.5×10^6	$300 \times 150 \times 300$	29.7

Table 2: *Particle counts, grid resolutions and simulation times per frame for each of our examples. Simulations were performed on a 16-core Xeon E5-268 2.67GHz machine.*

7 Discussion and Limitations

MPM. While MPM yields automatic collision and topology changes, it incurs some difficulties. For example, the grid introduces numerical plasticity, and it is difficult to represent sharp interfaces between materials. One down-side of our cubic interpolant is that we have a wider stencil compared to what most incompressible FLIP solvers use. This leads to additional numerical viscosity as well as increased computational expense. While it is tempting to use quadratic B-splines for rasterization paired with trilinear interpolation, low-order interpolation with MPM is known to have stability problems [Steffen et al. 2008]. Additionally, for this paper we focused on sticky boundaries because the materials we were simulating were typically sticky. Thus, deriving a free-slip boundary condition would be interesting future work. It would be interesting to consider alternative integration strategies that would yield bigger time steps, though our time steps tend to be commensurate with [Stomakhin et al. 2013].

Projection. Although the projection-like decoupling of pressure from MPM discretized deviatoric terms is valid away from the boundary, there is still coupling through the free surface boundary condition $\boldsymbol{\sigma} \cdot \boldsymbol{n} = \boldsymbol{\sigma}_\mu \cdot \boldsymbol{n} - p\boldsymbol{n} = \boldsymbol{0}$. In order to separate the MPM-based solution of the deviatoric terms from the pressure equations, we implicitly assume $\boldsymbol{\sigma}_\mu \cdot \boldsymbol{n} = \boldsymbol{0}$ during the MPM solve and $p = 0$ during the projection (at the surface). While this does guarantee that $\boldsymbol{\sigma} \cdot \boldsymbol{n} = \boldsymbol{\sigma}_\mu \cdot \boldsymbol{n} - p\boldsymbol{n} = \boldsymbol{0}$, it removes some flexibility as it is akin to enforcing $a + b = 0$ with $a = 0$ and $b = 0$. Note that the boundary condition $\boldsymbol{\sigma}_\mu \cdot \boldsymbol{n} = \boldsymbol{0}$ is automatically enforced at the free surface with an MPM discretization since it is the "natural" boundary from the variational principle on which MPM is based. While this decoupling certainly causes errors in both pressure and the velocities (see e.g. [Hirt and Shannon 1968] for discussion), this simplification is commonly done in both computer graphics [Carlson et al. 2002; Goktekin et al. 2004; Rasmussen et al. 2004; Losasso et al. 2006b; Batty and Bridson 2008] and engineering [Harlow and Welch 1965].

Performance. Our implementation was parallelized and has shown good scaling results with increasing number of CPU cores. However, the performance still remains an issue. In particular, the grid rasterization step (including matrix-vector multiplication in the implicit MPM solve), constitutes a significant portion of runtime. In the future, we might consider acceleration via CPU SIMD or GPGPU techniques to improve the performance. Also, employing simulation level of detail techniques could reduce run times in areas where the particles have settled. Alternatively, Lagrangian techniques such as [Solenthaler et al. 2007] have achieved material variation and melting effects with less computational cost. Nevertheless, we believe our formulation remains interesting because it provides a theoretical unification between two popular algorithms while also allowing formalized constitutive modeling.

Sampling. Particle methods can suffer from poor sample quality under large deformation. Even though pure Lagrangian methods can avoid drift when returning to a rest configuration, under significant plastic deformation, conditioning, sample density, and ac-

Figure 10: *Bringing a hot fluid stream in contact with a cold solid produces compelling phase transition effects. The image demonstrates both: simulated particle view with temperature distribution (top) and the rendering of our final meshed geometry (bottom).* ©*Disney.*

curacy may degrade, requiring remeshing (see e.g. [Bargteil et al. 2007]) or resampling. While we note that in the presence of less liquid-like behavior, drift is less of an issue, we plan to experiment with resampling techniques in the future.

Rendering. While modeling and simulation is simplified with particle methods, obtaining high quality rendering becomes more challenging. Since MPM naturally produces a density rasterization, index-of-refraction matched volume renderers can sometimes be applied (e.g. for snow). For most of the materials in this paper, however, we needed to render an interface, thus we turned to meshing solutions. Such techniques are common for liquid rendering and typically involve rasterizing particles to a grid using some (usually spherically symmetric) basis function followed by grid smoothing, contouring and final surface smoothing. These steps typically require per-example tuning and it is often impossible to recover as much detail as the particles seem to possess. This can be seen in Figure 10. We also experimented with anisotropic kernel techniques such as [Yu and Turk 2010], but we found that while they are very successful for liquids with visible surface tension, in our case they created more artifacts than they removed. Thus, any techniques that improve meshing will improve the final quality of our results.

8 Conclusion

In summary, we introduced a novel material point method for melting and solidifying materials using a heat solver to capture the underlying thermodynamics and alter mechanical parameters. The method is implicit and capable of simulating nearly incompressible materials using a Chorin-like projection solve. Hence, we have widened the range of materials MPM can handle, and we have demonstrated this span with several compelling melting and solidifying examples.

Acknowledgments

UCLA authors were partially supported by NSF (DMS-0502315, DMS-0652427, CCF-0830554, DOE (09-LR-04-116741-BERA), ONR (N000140310071, N000141010730, N000141210834), Intel STC-Visual Computing Grant (20112360) as well as a gift from Disney Research. We also appreciate the support at the studio from D. Meltzer, R. Sharma, D. Candela, and A. Hendrickson. Images rendered using Disney's Hyperion Renderer.

References

BARGTEIL, A. W., WOJTAN, C., HODGINS, J. K., AND TURK, G. 2007. A finite element method for animating large viscoplastic flow. *ACM Trans. Graph. 26*, 3.

BATTY, C., AND BRIDSON, R. 2008. Accurate viscous free surfaces for buckling, coiling, and rotating liquids. In *Proc 2008 ACM/Eurographics Symp Comp Anim*, 219–228.

BECKER, M., IHMSEN, M., AND TESCHNER, M. 2009. Corotated sph for deformable solids. In *Eurographics Conf. Nat. Phen.*, 27–34.

BONET, J., AND WOOD, R. 1997. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press.

CARLSON, M., MUCHA, P. J., VAN HORN, III, R. B., AND TURK, G. 2002. Melting and flowing. In *ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 167–174.

CARLSON, M., MUCHA, P., AND TURK, G. 2004. Rigid fluid: animating the interplay between rigid bodies and fluid. In *ACM Trans. on Graph.*, vol. 23, 377–384.

CHANG, Y., BAO, K., LIU, Y., ZHU, J., AND WU, E. 2009. A particle-based method for viscoelastic fluids animation. In *ACM Symp. Virt. Real. Soft. Tech.*, 111–117.

CHENTANEZ, N., GOKTEKIN, T. G., FELDMAN, B. E., AND O'BRIEN, J. F. 2006. Simultaneous coupling of fluids and deformable bodies. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 83–89.

CHOI, S.-C. T. 2006. *Iterative Methods for Singular Linear Equations and Least-Squares Problems*. PhD thesis, ICME, Stanford University, CA.

CHORIN, A. 1968. Numerical solution of the Navier-Stokes Equations. *Math. Comp. 22*, 745–762.

CLAUSEN, P., WICKE, M., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2013. Simulating liquids and solid-liquid interactions with lagrangian meshes. *ACM Trans. Graph. 32*, 2, 17:1–17:15.

DAGENAIS, F., GAGNON, J., AND PAQUETTE, E. 2012. A prediction-correction approach for stable sph fluid simulation from liquid to rigid. In *Proc. of Comp. Graph. Intl.*

DESBRUN, M., AND GASCUEL, M.-P. 1996. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Eurographics Workshop Comp. Anim. Sim.*, 61–76.

GOKTEKIN, T. G., BARGTEIL, A. W., AND O'BRIEN, J. F. 2004. A method for animating viscoelastic fluids. *ACM Trans. Graph. 23*, 3, 463–468.

GONZALEZ, O., AND STUART, A. 2008. *A First Course in Continuum Mechanics*. Cambridge texts in applied mathematics. Cambridge University Press.

HARLOW, F., AND WELCH, E. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys Fl 8*, 2182.

HIRT, C., AND SHANNON, J. 1968. Free-surface stress conditions for incompressible-flow calculations. *JCP 2*, 4, 403–411.

IRVING, G., TERAN, J., AND FEDKIW, R. 2004. Invertible finite elements for robust simulation of large deformation. In *Proc. 2004 ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 131–140.

IWASAKI, K., UCHIDA, H., DOBASHI, Y., AND NISHITA, T. 2010. Fast particle-based visual simulation of ice melting. *Comp. Graph. Forum 29*, 7, 2215–2223.

KEISER, R., ADAMS, B., GASSER, D., BAZZI, P., DUTRÉ, P., AND GROSS, M. 2005. A unified lagrangian approach to solid-fluid animation. In *Eurographics/IEEE VGTC Conf. Point-Based Graph.*, 125–133.

KIM, T., ADALSTEINSSON, D., AND LIN, M. C. 2006. Modeling ice dynamics as a thin-film stefan problem. In *Proc 2006 ACM SIGGRAPH/Eurographics Symp Comp Anim*, 167–176.

KWATRA, N., SU, J., GRETARSSON, J., AND FEDKIW, R. 2009. A method for avoiding the acoustic time-step restriction in compressible flow. *J. Comp. Phys. 228*, 4146–4161.

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

LII, S.-Y., AND WONG, S.-K. 2013. Ice melting simulation with water flow handling. *Vis. Comp.*, 1–8.

LOSASSO, F., IRVING, G., GUENDELMAN, E., AND FEDKIW, R. 2006. Melting and burning solids into liquids and gases. *IEEE Trans. Vis. Comp. Graph. 12*, 343–352.

LOSASSO, F., SHINAR, T., SELLE, A., AND FEDKIW, R. 2006. Multiple interacting liquids. *ACM Trans. Graph. 25*, 3, 812–819.

MARÉCHAL, N., GUÉRIN, E., GALIN, E., MÉRILLOU, S., AND MÉRILLOU, N. 2010. Heat transfer simulation for modeling realistic winter sceneries. *Comp. Graph. Forum 29*, 2, 449–458.

MARTIN, S., KAUFMANN, P., BOTSCH, M., GRINSPUN, E., AND GROSS, M. 2010. Unified simulation of elastic rods, shells, and solids. *ACM Trans. Graph. 29*, 4 (July), 39:1–39:10.

MAST, C., MACKENZIE-HELNWEIN, P., ARDUINO, P., MILLER, G., AND SHIN, W. 2012. Mitigating kinematic locking in the material point method. *J. Comp. Phys. 231*, 16, 5351–5373.

MONAGHAN, J. J. 1992. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics 30*, 543–574.

MÜLLER, M., KEISER, R., NEALEN, A., PAULY, M., GROSS, M., AND ALEXA, M. 2004. Point based animation of elastic, plastic and melting objects. In *ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 141–151.

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3, 471–478.

PAIVA, A., PETRONETTO, F., LEWINER, T., AND TAVARES, G. 2006. Particle-based non-newtonian fluid animation for melting objects. In *Conf. Graph. Patt. Images*, 78–85.

PAIVA, A., PETRONETTO, F., LEWINER, T., AND TAVARES, G. 2009. Particle-based viscoplastic fluid/solid simulation. *Comp. Aided Des. 41*, 4, 306–314.

RASMUSSEN, N., ENRIGHT, D., NGUYEN, D., MARINO, S., SUMNER, N., GEIGER, W., HOON, S., AND FEDKIW, R. 2004. Directable photorealistic liquids. In *ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 193–202.

ROBINSON-MOSHER, A., SHINAR, T., GRETARSSON, J., SU, J., AND FEDKIW, R. 2008. Two-way coupling of fluids to rigid and deformable solids and shells. *ACM Trans. Graph. 27*, 3 (Aug.), 46:1–46:9.

SERWAY, R. A., AND JEWETT, J. W. 2009. *Physics for Scientists and Engineers*. Cengage Learning.

SOLENTHALER, B., AND PAJAROLA, R. 2009. Predictive-corrective incompressible sph. In *ACM transactions on graphics (TOG)*, vol. 28, ACM, 40.

SOLENTHALER, B., SCHLÄFLI, J., AND PAJAROLA, R. 2007. A unified particle model for fluid-solid interactions: Research articles. *Comp. Anim. Virt. Worlds 18*, 1, 69–82.

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

STOMAKHIN, A., HOWES, R., SCHROEDER, C., AND TERAN, J. 2012. Energetically consistent invertible elasticity. In *ACM SIGGRAPH/Eurographics Symp. Comp. Anim.*, 25–32.

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

STORA, D., AGLIATI, P.-O., CANI, M.-P., NEYRET, F., AND GASCUEL, J.-D. 1999. Animating lava flows. In *Graph. Int.*, 203–210.

SULSKY, D., ZHOU, S.-J., AND SCHREYER, H. 1995. Application of particle-in-cell method to solid mechanics. *Comp. Phys. Comm. 87*, 236–252.

TERZOPOULOS, D., PLATT, J., AND FLEISCHER, K. 1991. Heating and melting deformable models. *J. Vis. Comp. Anim. 2*, 2, 68–73.

TESCHNER, M., HEIDELBERGER, B., MULLER, M., AND GROSS, M. 2004. A versatile and robust model for geometrically complex deformable solids. In *Comp. Graph. Int.*, 312–319.

WEI, X., LI, W., AND KAUFMAN, A. 2003. Melting and flowing of viscous volumes. In *Intl. Conf. Comp. Anim. Social Agents*, 54–60.

WICKE, M., RITCHIE, D., KLINGNER, B. M., BURKE, S., SHEWCHUK, J. R., AND O'BRIEN, J. F. 2010. Dynamic local remeshing for elastoplastic simulation. *ACM Transactions on Graphics 29*, 4 (July), 49:1–11. Proc. of ACM SIGGRAPH 2010.

WOJTAN, C., AND TURK, G. 2008. Fast viscoelastic behavior with thin features. *ACM Trans. Graph. 27*, 3, 47:1–47:8.

WOJTAN, C., CARLSON, M., MUCHA, P. J., AND TURK, G. 2007. Animating corrosion and erosion. In *Eurographics Conf. Nat. Phen.*, 15–22.

WOJTAN, C., THÜREY, N., GROSS, M., AND TURK, G. 2009. Deforming meshes that split and merge. *ACM Trans. Graph. 28*, 3, 76:1–76:10.

YU, J., AND TURK, G. 2010. Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *Proc. of the 2010 ACM SIGGRAPH/Eurographics Symp. on Comp. Anim.*, Eurographics Association, 217–225.

ZHAO, Y., WANG, L., QIU, F., KAUFMAN, A., AND MUELLER, K. 2006. Melting and flowing in multiphase environment. *Comp. Graph. 30*, 2006.

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