University of California

Los Angeles

Modeling and Simulation of Thermomechanical Elasto-viscoplastic Material and Ductile Fracture with the Material Point Method

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Mathematics

by

Mengyuan Ding

2020

Abstract of the Dissertation

Modeling and Simulation of Thermomechanical Elasto-viscoplastic Material and Ductile Fracture with the Material Point Method

by

Mengyuan Ding
Doctor of Philosophy in Mathematics
University of California, Los Angeles, 2020
Professor Joseph M. Teran, Chair

This dissertation focuses on applications and extensions of the Material Point Method(MPM) in simulating ductile fracture and thermomechanical material behavior for baking and cooking. We conclude the two major contributions as follows:

First, we present novel techniques for simulating and visualizing ductile fracture with MPM. We utilize traditional particle-based MPM [SZS95] as well as the Lagrangian energy formulation of [JSS15] that formulates the deformation gradient and potential energy through a tetrahedron mesh. We model failure and fracture via elasto-plasticity with damage. Material shows elastic behavior until the deformation exceeds a Rankine or von Mises yield criterion, at which point irreversible damage starts to occur. We introduce a softening model that shrinks the yield surface until a damage threshold is reached. Once damaged the material Lamé coefficients are modified to represent failure. We design visualization techniques for rendering the boundary of the material and its intersections with evolving crack surfaces. Our approach uses a simple and efficient element splitting strategy for tetrahedron meshes to represent crack surfaces. For traditional particle-based MPM we use an initial Delaunay tetrahedralization to connect randomly initialized MPM particles and form a reference mesh. Our visualization technique is a postprocess and can be run separately after the MPM simulation for efficiency. We demonstrate the strength of our method with a number of challenging simulations of ductile failure with considerable and persistent self contact.

Our second contribution is a Material Point Method for visual simulation of baking breads, cookies, pancakes and similar materials. We develop a novel thermomechanical model using mixture theory to resolve interactions between individual water, gas and solid species. Heat transfer with thermal expansion is used to model thermal variations in material properties. Water based mass transfer is resolved through the porous mixture. Gas represents carbon dioxide produced by leavening agents in the baking process, and the solid component is modeled as a visco-elastoplastic material to represent its varied and complex rheological properties. Water content in the mixture reduces during the baking process according to Fick's Law which contributes to the drying and cracking of crust at the material boundary. Carbon dioxide gas produced by leavening agents during baking creates internal pressure that causes rising. The visco-elastoplastic model for the dough is temperature dependent and is used to model melting and solidification. We discretize the governing equations using a novel Material Point Method designed to track the solid phase of the mixture.

The dissertation of Mengyuan Ding is approved.

Jeffrey D. Eldredge

Christopher R. Anderson

Luminita Aura Vese

Joseph M. Teran, Committee Chair

University of California, Los Angeles

2020

To Haru,

the warmest fuzzball in the world, who diligently sat behind the computer and supervised the completion of this work.

Table of Contents

List of Figures

List of Tables

Acknowledgments

My greatest gratitude and appreciation goes to my advisor, Professor Joseph Teran, for his continued guidance and dedicated support of my research. I would also like to thank Professor Christopher Anderson, Professor Jeffrey Eldredge and Professor Luminita Vese for serving on my doctoral committee and helping me through my PhD studies.

I am very grateful to Professor Inwon Kim, Professor Wotao Yin, Professor Jiann-Wen Ju, Professor Andrea Bertozzi and all my teachers at UCLA for their inspiring lectures and valuable advice through the years.

I thank my labmates for the countless days and nights we worked together. I owe special thanks to Ted Gast for being so kind and knowledgeable, and never impatient when I have the most basic and silly questions; Xuchen Han and Qi Guo for working alongside of me throughout the course of my degree, giving valuable suggestions and sharing their experiences; Andre Pradhana for his kindness and helpful advisement during my internship at DreamWorks, giving me the chance to look at research from a different perspective. All of them have been great mentors, trusting companions and wonderful friends.

I have to thank my fiancé and our three cats for their utmost trust, friendship and love. They have always kept me company when I work late into the night, and have been one of the biggest sources of warmth and joy in my life. They are more perfect than I could ever dream of.

Finally I want to thank my family for their unconditional love and support throughout my life. I thank my mom for everything she has done for me and for never losing faith in me even when I do myself; my dad for being my biggest role model and for fostering my interest and passion in mathematics from the very beginning. I've been following in his footsteps all this time.

This dissertation contains content from previously published work. Chapter 2 from [WDG19] and Chapter 3 from [DHW19].

Vita

2015	B.S.(Applied Mathematics), University of Science and Technology of China
2015 - 2020	Teaching Assistant, Mathematics Department, UCLA
2017 - 2020	Research Assistant, Mathematics Department, UCLA
2019.6 - 2019.9	Research Scientist Intern, Forma Technologies Inc.
2019.9 - 2019.12	FX-R&D Intern, DreamWorks Animation
2020.2 - 2020.6	Software Developer, Autodesk

Publications

S. Wang, M. Ding, T.F. Gast, L. Zhu, S. Gagniere, C. Jiang, and J.M. Teran. "Simulation and Visualization of Ductile Fracture with the Material Point Method." Proceedings of the ACM on Computer Graphics and Interactive Techniques, 2019

M. Ding, X. Han, S. Wang, T.F. Gast, and J.M. Teran. "A Thermomechanical Material Point Method for Baking and Cooking." ACM Transactions on Graphics (TOG), 2019

CHAPTER 1

Introduction and Theoretical Background

1.1 Continuum Mechanics

In this section we briefly discuss the fundamental continuum assumptions on which we build our research. The definitions and derivations closely follow [GS08].

1.1.1 Continuum Bodies and Kinematics

We first make a most basic assumption that the material involved can be modeled as a continuum. This allows us to identify a material body at any fixed instant of time with an open subset B of Euclidean space $\mathbb{E}^3$. Each material particle is identified with a point in B.

The motion of a body with reference configuration B_0 is described by a continuous map $\phi : B_0 \times [0, \infty) \to \mathbb{E}^3$. At any time t, each point $\mathbf{X}$ in the reference configuration B_0 is mapped to a point $\mathbf{x}(t) = \phi(\mathbf{X}, t)$ in the current configuration B_t. The continuity assumption further implies the existence of an inverse deformation map $\phi^{-1} : B_t \to B_0$ such that

$$\phi^{-1}(\phi(\mathbf{X}, t), t) = \mathbf{X}, \quad \phi(\phi^{-1}(\mathbf{x}, t), t) = \mathbf{x}$$

The material, or Lagrangian, velocity and acceleration of the material particle $\mathbf{X}$ are denoted as

$$\mathbf{V}(\mathbf{X}, t) = \frac{\partial \phi}{\partial t}(\mathbf{X}, t)$$
$$\mathbf{A}(\mathbf{X}, t) = \frac{\partial \mathbf{V}}{\partial t}(\mathbf{X}, t) = \frac{\partial^2 \phi}{\partial t^2}(\mathbf{X}, t)$$

The corresponding spatial description, or Eulerian form, of the velocity and acceleration fields are defined as

$$\mathbf{v}(\mathbf{x},t) = \mathbf{V}(\phi^{-1}(\mathbf{x},t),t), \qquad \mathbf{V}(\mathbf{X},t) = \mathbf{v}(\phi(\mathbf{X},t),t)$$
$$\mathbf{a}(\mathbf{x},t) = \mathbf{A}(\phi^{-1}(\mathbf{x},t),t), \qquad \mathbf{A}(\mathbf{X},t) = \mathbf{a}(\phi(\mathbf{X},t),t)$$

We then have the following relation:

$$\begin{aligned}
\mathbf{A}(\mathbf{X},t) &= \frac{\partial \mathbf{V}}{\partial t}(\mathbf{X},t) \\
&= \frac{\partial}{\partial t}\mathbf{v}(\phi(\mathbf{X},t),t) \\
&= \frac{\partial \mathbf{v}}{\partial t}(\phi(\mathbf{X},t),t) + \frac{\partial \mathbf{v}}{\partial \mathbf{x}}(\phi(\mathbf{X},t),t)\cdot\frac{\partial \phi}{\partial t}(\mathbf{X},t) \\
&= \frac{\partial \mathbf{v}}{\partial t}(\phi(\mathbf{X},t),t) + \frac{\partial \mathbf{v}}{\partial \mathbf{x}}(\phi(\mathbf{X},t),t)\cdot\mathbf{v}(\phi(\mathbf{X},t),t) \\
\Rightarrow \mathbf{a}(\mathbf{x},t) &= \mathbf{A}(\phi^{-1}(\mathbf{x},t),t) \\
&= \frac{\partial \mathbf{v}}{\partial t}(\mathbf{x},t) + \frac{\partial \mathbf{v}}{\partial \mathbf{x}}(\mathbf{x},t)\cdot\mathbf{v}(\mathbf{x},t) \\
&:= \frac{D}{Dt}\mathbf{v}(\mathbf{x},t)
\end{aligned}$$

Here we define the material derivative $\frac{D}{Dt}$ as $\frac{D*}{Dt} = \frac{\partial *}{\partial t} + \frac{\partial *}{\partial \mathbf{x}}\cdot\mathbf{v}$.

The deformation gradient $\mathbf{F}(\mathbf{X},t) = \frac{\partial \phi}{\partial \mathbf{X}}(\mathbf{X},t) = \frac{\partial \mathbf{x}}{\partial \mathbf{X}}(\mathbf{X},t)$ quantifies the strain and provides information on the local behavior of the flow map ϕ. In particular, we have the following linear approximation of ϕ for any $\tilde{\mathbf{X}}$ near a given point $\mathbf{X}$:

$$\phi(\tilde{\mathbf{X}},t) \approx \nabla^{\mathbf{X}}\phi(\mathbf{X},t)(\tilde{\mathbf{X}}-\mathbf{X}) + \phi(\mathbf{X},t) = \mathbf{F}(\mathbf{X},t)(\tilde{\mathbf{X}}-\mathbf{X}) + \mathbf{x}$$

We denote $J(\mathbf{X},t) = \det \mathbf{F}(\mathbf{X},t)$.

1.1.2 Balance Laws

Consider a continuum body with reference configuration B_0 undergoing a motion ϕ, and B_t denotes the configuration at any given time t. The mass density of the body is denoted as

$\rho(\mathbf{x}, t)$, then for any arbitrary open subset Ω_t of B_t the mass and linear momentum in the region are given by

$$\text{mass}[\Omega_t] = \int_{\Omega_t} \rho(\mathbf{x}, t) d\mathbf{x} \tag{1.1}$$

$$\text{momentum}[\Omega_t] = \int_{\Omega_t} \rho(\mathbf{x}, t)\mathbf{v}(\mathbf{x}, t) d\mathbf{x} \tag{1.2}$$

The material description of the mass density is denoted as $R(\mathbf{X}, t)$, with the relation $\rho(\phi(\mathbf{X}, t), t) = R(\mathbf{X}, t)$ similar as before.

1.1.2.1 Conservation of Mass

The conservation law of mass states that the mass of any open subset of a continuum body does not change as the body changes place and shape, that is

$$\frac{d}{dt}\text{mass}[\Omega_t] = 0, \qquad \forall \Omega_t \subseteq B_t$$

This implies that $\text{mass}[\Omega_t] = \text{mass}[\Omega_0]$ where Ω_0 stands for the corresponding subset in the reference configuration. We have

$$\begin{aligned} \text{mass}[\Omega_t] &= \int_{\Omega_t} \rho(\mathbf{x}, t) d\mathbf{x} \\ &= \int_{\Omega_0} \rho(\phi(\mathbf{X}, t), t) \det \mathbf{F}(\mathbf{X}, t) d\mathbf{X} \\ &= \int_{\Omega_0} R(\mathbf{X}, t) J(\mathbf{X}, t) d\mathbf{X} \end{aligned}$$

On the other hand $\text{mass}[\Omega_0] = \int_{\Omega_0} R(\mathbf{X}, 0) d\mathbf{X}$. By the Localization Theorem we deduce the conservation of mass equation in Lagrangian form

$$R(\mathbf{X}, 0) = R(\mathbf{X}, t) J(\mathbf{X}, t), \qquad \forall \mathbf{X} \in B_0, \quad t \geq 0. \tag{1.3}$$

Taking the time derivative of both sides and using $\frac{\partial J}{\partial t} = J\nabla^{\mathbf{x}} \cdot \mathbf{v}(\phi^{-1}(\mathbf{X}, t))$ we get

$$
\begin{aligned}
0 &= \frac{\partial}{\partial t}(R(\mathbf{X}, t)J(\mathbf{X}, t)) \\
&= \frac{\partial}{\partial t}\rho(\mathbf{x}, t)J(\mathbf{X}, t) + \rho(\mathbf{x}, t)J(\mathbf{X}, t)\nabla^{\mathbf{x}} \cdot \mathbf{v}(\mathbf{x}, t) \\
&= (\frac{\partial \rho}{\partial t}(\mathbf{x}, t) + \frac{\partial \rho}{\partial \mathbf{x}}(\mathbf{x}, t) \cdot \mathbf{v}(\mathbf{x}, t))J(\phi^{-1}(\mathbf{x}, t), t) + \rho(\mathbf{x}, t)\nabla^{\mathbf{x}} \cdot \mathbf{v}(\mathbf{x}, t)J(\phi^{-1}(\mathbf{x}, t), t) \\
&= (\frac{D\rho}{Dt} + \rho\nabla^{\mathbf{x}} \cdot \mathbf{v})(\mathbf{x}, t)J(\phi^{-1}(\mathbf{x}, t), t)
\end{aligned}
$$

For any admissible motion we should have $J > 0$, therefore we arrive at the Eulerian conservation of mass equation:

$$
\frac{D\rho}{Dt} + \rho\nabla^{\mathbf{x}} \cdot \mathbf{v} = 0, \qquad \forall \mathbf{x} \in B_t, \quad t \geq 0. \tag{1.4}
$$

1.1.2.2 Conservation of Linear Momentum

Similar as before the conservation of linear momentum is stated as

$$
\frac{d}{dt}\text{momentum}[\Omega_t] = \int_{\partial\Omega_t} \mathbf{t}(\mathbf{x}, t)ds + \int_{\Omega_t} \rho(\mathbf{x}, t)\mathbf{b}(\mathbf{x}, t)d\mathbf{x}, \qquad \forall \Omega_t \subseteq B_t
$$

where $\mathbf{t}(\mathbf{x}, t)$ and $\mathbf{b}(\mathbf{x}, t)$ stands for the traction and body force field respectively. The traction field is determined by Cauchy stress $\mathbf{t} = \boldsymbol{\sigma}\mathbf{n}$ where $\mathbf{n}$ is the outward unit normal. Then by Divergence Theorem we have

$$
\text{RHS} = \int_{\partial\Omega_t} \boldsymbol{\sigma}\mathbf{n}ds + \int_{\Omega_t} \rho\mathbf{b}d\mathbf{x} = \int_{\Omega_t} (\nabla^{\mathbf{x}} \cdot \boldsymbol{\sigma} + \rho\mathbf{b})d\mathbf{x}
$$

Combined with

$$
\begin{aligned}
\text{LHS} = \frac{d}{dt}\int_{\Omega_t} \rho(\mathbf{x}, t)\mathbf{v}(\mathbf{x}, t)d\mathbf{x} &= \frac{d}{dt}\int_{\Omega_0} R(\mathbf{X}, t)\mathbf{V}(\mathbf{X}, t)J(\mathbf{X}, t)d\mathbf{X} \\
&= \frac{d}{dt}\int_{\Omega_0} R(\mathbf{X}, 0)\mathbf{V}(\mathbf{X}, t)d\mathbf{X} \\
&= \int_{\Omega_0} R(\mathbf{X}, 0)\frac{\partial}{\partial t}\mathbf{V}(\mathbf{X}, t)d\mathbf{X} \\
&= \int_{\Omega_t} \rho(\mathbf{x}, t)\frac{\partial}{\partial t}\mathbf{v}(\mathbf{x}(t), t)d\mathbf{x} \\
&= \int_{\Omega_t} \rho\frac{D\mathbf{v}}{Dt}d\mathbf{x}
\end{aligned}
$$

And again by Localization Theorem we get the Eulerian conservation of linear momentum:

$$\rho \frac{D\mathbf{v}}{Dt} = \nabla^{\mathbf{x}} \cdot \boldsymbol{\sigma} + \rho \mathbf{b}, \qquad \forall \mathbf{x} \in B_t, \quad t \geq 0. \tag{1.5}$$

Going back to the integral forms above we have

$$\begin{aligned}
&\int_{\Omega_0} R(\mathbf{X},0)\frac{\partial}{\partial t}\mathbf{V}(\mathbf{X},t)d\mathbf{X} \\
&= \int_{\partial\Omega_t} \boldsymbol{\sigma}(\mathbf{x},t)\mathbf{n}(\mathbf{x},t)ds + \int_{\Omega_t} \rho(\mathbf{x},t)\mathbf{b}(\mathbf{x},t)d\mathbf{x} \\
&= \int_{\partial\Omega_0} J(\mathbf{X},t)\boldsymbol{\sigma}(\phi(\mathbf{X},t),t)\mathbf{F}^{-T}(\mathbf{X},t)\mathbf{N}(\mathbf{X})d\mathbf{X} + \int_{\Omega_0} J(\mathbf{X},t)R(\mathbf{X},t)\mathbf{b}(\phi(\mathbf{X},t),t)d\mathbf{X} \\
&= \int_{\partial\Omega_0} \mathbf{P}(\mathbf{X},t)\mathbf{N}(\mathbf{X})d\mathbf{X} + \int_{\Omega_0} R(\mathbf{X},0)\mathbf{b}_m(\mathbf{X},t)d\mathbf{X} \\
&= \int_{\Omega_0} (\nabla^{\mathbf{X}} \cdot \mathbf{P}(\mathbf{X},t) + R(\mathbf{X},0)\mathbf{b}_m(\mathbf{X},t))d\mathbf{X}
\end{aligned}$$

where $\mathbf{P} = J\boldsymbol{\sigma}\mathbf{F}^{-T}$ is the first Piola-Kirchhoff stress, and $\mathbf{b}_m$ stands for the material description of the spatial body force field $\mathbf{b}$. Again by Localization Theorem we arrive at the Lagrangian conservation of linear momentum:

$$R(\mathbf{X},0)\frac{\partial^2\phi}{\partial t^2}(\mathbf{X},t) = R(\mathbf{X},0)\frac{\partial \mathbf{V}}{\partial t}(\mathbf{X},t) = \nabla^{\mathbf{X}} \cdot \mathbf{P}(\mathbf{X},t) + R(\mathbf{X},0)\mathbf{b}_m(\mathbf{X},t), \quad \forall \mathbf{X} \in B_0, \quad t \geq 0. \tag{1.6}$$

1.1.2.3 First and Second Laws of Thermodynamics

The Lagrangian form of the laws of Thermodynamics are given below:([GS08])

- (First Law of Thermodynamics) Energy Balance

$$\dot{\Phi} = \mathbf{P} : \dot{\mathbf{F}} - \nabla \cdot \mathbf{Q} + R_0(\mathbf{X})r_m(\mathbf{X},t) \tag{1.7}$$

- (Second Law of Thermodynamics) Clausius-Duhem Inequality

$$\Theta\dot{\eta}_m \geq R_0 r_m - \nabla \cdot \mathbf{Q} + \Theta^{-1}(\nabla\Theta \cdot \mathbf{Q}) \tag{1.8}$$

Where the dot symbol stands for the time derivative, Φ is the internal energy density per unit volume, $\mathbf{P}$ is the first Piola-Kirchhoff stress, $\mathbf{F}$ is the deformation gradient, $\mathbf{Q}(\mathbf{X},t)$ is the material heat flux field defined through the Fourier-Stokes heat flux $\mathbf{q}(\mathbf{x}, t)$ by $\mathbf{Q} = J\mathbf{F}^{-1}\mathbf{q}$ with $J = \det(\mathbf{F})$, $R_0(\mathbf{X}) = R(\mathbf{X}, 0)$ is the material density at rest configuration, r_m is body heating supply, Θ is the absolute temperature, and η_m is the entropy density per unit volume. The free energy density is defined as

$$\Psi(\mathbf{X}, t) = \Phi(\mathbf{X}, t) - \Theta(\mathbf{X}, t)\eta_m(\mathbf{X}, t)$$

For thermoelastic solids we have the following assumptions:

- The functions Φ, Ψ and η_m can be represented as functions of $\mathbf{F}$ and Θ. Furthermore we have
$$\frac{\partial \Phi}{\partial \Theta} > 0, \qquad \frac{\partial \eta_m}{\partial \Theta} > 0$$
for any $\mathbf{F}$ with $\det(\mathbf{F}) > 0$ and $\Theta > 0$.

- The material heat flux vector $\mathbf{Q}$ is related to $\mathbf{F}$ and Θ by
$$\mathbf{Q} = -\mathbf{K}(\mathbf{F}, \Theta)\nabla\Theta$$
where $\mathbf{K}$ is called the thermal conductivity function, and the gradient is with respect to $\mathbf{X}$.

These assumptions combined with the two laws yield:

$$\begin{aligned}\dot{\Psi} &= \dot{\Phi} - \dot{\Theta}\eta_m - \Theta\dot{\eta}_m \\ &= \mathbf{P} : \dot{\mathbf{F}} + (R_0 r_m - \nabla \cdot \mathbf{Q}) - \dot{\Theta}\eta_m - \Theta\dot{\eta}_m \\ &\leq \mathbf{P} : \dot{\mathbf{F}} + (\Theta\dot{\eta}_m - \Theta^{-1}(\nabla\Theta \cdot \mathbf{Q})) - \dot{\Theta}\eta_m - \Theta\dot{\eta}_m \\ &= \mathbf{P} : \dot{\mathbf{F}} + \Theta^{-1}(\nabla\Theta \cdot \mathbf{K}\nabla\Theta) - \dot{\Theta}\eta_m.\end{aligned}$$

Since $\Psi = \Psi(\mathbf{F}, \Theta)$ we can write

$$\dot{\Psi} = \frac{\partial \Psi}{\partial \mathbf{F}} : \dot{\mathbf{F}} + \frac{\partial \Psi}{\partial \Theta}\dot{\Theta}.$$

Then the above equation becomes

$$\left(\frac{\partial \Psi}{\partial \mathbf{F}} - \mathbf{P}\right) : \dot{\mathbf{F}} + \left(\eta_m + \frac{\partial \Psi}{\partial \Theta}\right)\dot{\Theta} - \Theta^{-1}(\nabla\Theta \cdot \mathbf{K}\nabla\Theta) \leq 0 \tag{1.9}$$

Since $\mathbf{F}$, Θ and their derivatives are independent of each other, we must have $\mathbf{P} = \frac{\partial \Psi}{\partial \mathbf{F}}$, $\eta_m = -\frac{\partial \Psi}{\partial \Theta}$, and $\mathbf{K}$ is positive semi-definite. Note that the first equation coincides with the definition of the first Piola-Kirchhoff stress. In this work for simplicity we take $\mathbf{q} = -\kappa \nabla \theta$, with a material heat conductivity constant $\kappa > 0$. With this formulation of $\mathbf{q}$ we have

$$\mathbf{Q} = J\mathbf{F}^{-1}\mathbf{q} = -\kappa J\mathbf{F}^{-1}\nabla^{\mathbf{x}}\theta = -\kappa J\mathbf{F}^{-1}\mathbf{F}^{-T}\nabla^{\mathbf{X}}\Theta.$$

So $\mathbf{K} = \kappa J\mathbf{F}^{-1}\mathbf{F}^{-T}$ satisfies the positive semi-definite constraint.

We define the specific heat at constant volume α as

$$\begin{aligned} 0 < \alpha &= \frac{1}{R_0(X)}\frac{\partial \Phi}{\partial \Theta} = \frac{1}{R_0}(\frac{\partial \Psi}{\partial \Theta} + \eta_m + \Theta\frac{\partial \eta_m}{\partial \Theta}) \\ &= \frac{1}{R_0}\Theta\frac{\partial \eta_m}{\partial \Theta} = -\frac{1}{R_0}\Theta\frac{\partial^2 \Psi}{\partial \Theta^2}. \end{aligned}$$

Note that then we need $\frac{\partial^2 \Psi}{\partial \Theta^2} < 0$.

In multiplicative plasticity theory the deformation gradient $\mathbf{F}$ is separated into the elastic component $\mathbf{F}^E$ and the plastic (or viscoplastic) part $\mathbf{F}^P$, so we have $\dot{\mathbf{F}} = \dot{\mathbf{F}}^E\mathbf{F}^P + \mathbf{F}^E\dot{\mathbf{F}}^P$, which yields

$$\dot{\mathbf{F}}^E = (\dot{\mathbf{F}} - \mathbf{F}^E\dot{\mathbf{F}}^P)(\mathbf{F}^P)^{-1}$$

Now we view Ψ as a function of $\mathbf{F}^E$ and Θ. Following the same steps as before we can derive a similar inequality as Equation (1.9):

$$\begin{aligned} &\left(\frac{\partial \Psi}{\partial \mathbf{F}^E}(\mathbf{F}^P)^{-T} - \mathbf{P}\right) : \dot{\mathbf{F}} + \left(\eta_m + \frac{\partial \Psi}{\partial \Theta}\right)\dot{\Theta} \\ &-\Theta^{-1}(\nabla\Theta \cdot \mathbf{K}\nabla\Theta) - \frac{\partial \Psi}{\partial \mathbf{F}^E}(\mathbf{F}^P)^{-T}(\mathbf{F}^E)^T : \dot{\mathbf{F}}^P \leq 0. \end{aligned}$$

Again we get $\mathbf{P} = \frac{\partial \Psi}{\partial \mathbf{F}^E}(\mathbf{F}^P)^{-T}$ which matches the definition of the first Piola-Kirchhoff stress, $\eta_m = -\frac{\partial \Psi}{\partial \Theta}$, and $\mathbf{K}$ is positive semi-definite. The last term stands for plastic energy dissipation, and we need:

$$\mathbf{P}(\mathbf{F}^E)^T : \dot{\mathbf{F}}^P \geq 0.$$

Going back to the energy balance equation, we can rewrite it as

$$\begin{aligned}
0 &= \dot{\Phi} - (\dot{\Psi} + \dot{\Theta}\eta_m + \Theta\dot{\eta}_m) \qquad (1.10)\\
&= (\mathbf{P} : \dot{\mathbf{F}} - \nabla \cdot \mathbf{Q} + R_0 r_m) - \left(\frac{\partial \Psi}{\partial \mathbf{F}} : \dot{\mathbf{F}} + \frac{\partial \Psi}{\partial \Theta}\dot{\Theta} + \dot{\Theta}\eta_m + \Theta\dot{\eta}_m\right)\\
&= (\mathbf{P} : \dot{\mathbf{F}} - \nabla \cdot \mathbf{Q} + R_0 r_m) - (\mathbf{P} : \dot{\mathbf{F}} - \eta_m\dot{\Theta} + \dot{\Theta}\eta_m + \Theta\dot{\eta}_m)\\
&= -\nabla \cdot \mathbf{Q} + R_0 r_m - \Theta\dot{\eta}_m\\
&= -\nabla \cdot \mathbf{Q} + R_0 r_m - \Theta\left(\frac{\partial \eta_m}{\partial \mathbf{F}} : \dot{\mathbf{F}} + \frac{\partial \eta_m}{\partial \Theta}\dot{\Theta}\right)\\
&= \nabla \cdot (\mathbf{K}\nabla\Theta) + R_0 r_m + \Theta\frac{\partial^2 \Psi}{\partial \mathbf{F}\partial\Theta} : \dot{\mathbf{F}} - R_0\alpha\dot{\Theta}.
\end{aligned}$$

We can further derive the Eulerian form of the balance equation using similar methods as before:

$$\begin{aligned}
0 &= \int_{\Omega_0}\left(\nabla^{\mathbf{X}} \cdot (\mathbf{K}\nabla^{\mathbf{X}}\Theta) + R_0 r_m(\mathbf{X}, t) + \Theta\frac{\partial^2 \Psi}{\partial \mathbf{F}\partial\Theta} : \dot{\mathbf{F}} - R_0\alpha\dot{\Theta}\right) d\mathbf{X}\\
&= \int_{\Omega_t}\left(\nabla^{\mathbf{x}} \cdot (\kappa\nabla^{\mathbf{x}}\theta(\mathbf{x}, t)) + \rho(\mathbf{x}, t)r(\mathbf{x}, t) + \theta(\mathbf{x}, t)\frac{1}{J}\frac{\partial^2 \psi}{\partial \mathbf{F}\partial\theta} : \dot{\mathbf{F}} - \rho\alpha\frac{D\theta}{Dt}\right) d\mathbf{x}
\end{aligned}$$

The equation holds for any region of interest. Therefore by Localization Theorem and also using

$$\frac{\partial \mathbf{F}}{\partial t}(\mathbf{X}, t) = \frac{\partial}{\partial t}\frac{\partial \boldsymbol{\phi}(\mathbf{X}, t)}{\partial \mathbf{X}} = \frac{\partial \mathbf{V}(\mathbf{X}, t)}{\partial \mathbf{X}} = \frac{\partial \mathbf{v}(\mathbf{x}, t)}{\partial \mathbf{x}}\frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \nabla\mathbf{v}\mathbf{F},$$

we get the Eulerian heat equation

$$\rho\alpha\frac{D\theta}{Dt} = \nabla \cdot (\kappa\nabla\theta) + \rho r + \frac{1}{J}\theta\frac{\partial^2 \psi}{\partial \mathbf{F}\partial\theta} : \nabla\mathbf{v}\mathbf{F} \qquad (1.11)$$

1.2 Material Point Method

The Material Point Method (MPM) views material deformation in both a Lagrangian and an Eulerian fashion. The material body is represented by a collection of material particles denoted p, which keeps track of mass m_p, velocity $\mathbf{v}_p$, position $\mathbf{x}_p$ and other physical attributes. The governing equations are solved on a background Eulerian grid where all stress based forces are computed. Therefore we need to transfer the material state to the Eulerian configuration to incorporate the effects of material forces; after the solve these effects are transferred back to the material particles which then get updated in the usual Lagrangian way.

Here we list the most basic steps in the MPM solver for each time step:

- Particle mass and momentum are transferred to grid nodes through interpolating functions defined over grid nodes $\mathbf{i}$, denoted $N_{\mathbf{i}}(\mathbf{x})$. Typically C_1 continuity is required for the interpolating functions to prevent cell-crossing instability. In this thesis quadratic B splines are used:

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

  Denote $w_{\mathbf{i}p} = N_{\mathbf{i}}(\mathbf{x}_p)$, the mass and linear momentum transfers are defined as

$$\begin{aligned} m_{\mathbf{i}} &= \sum_p m_p w_{\mathbf{i}p} \\ (m\mathbf{v})_{\mathbf{i}} &= \sum_p m_p \mathbf{v}_p w_{\mathbf{i}p} \\ \Rightarrow \mathbf{v}_{\mathbf{i}} &= \frac{(m\mathbf{v})_{\mathbf{i}}}{m_{\mathbf{i}}} = \frac{\sum_p m_p w_{\mathbf{i}p}}{\sum_p m_p \mathbf{v}_p w_{\mathbf{i}p}} \end{aligned}$$

  Note that these transfers preserve the total mass and linear momentum of all material particles versus all grid nodes.

- Grid momentum is updated in a variational way from the potential energy in the system. Chapter 2 discusses the difference in this step between standard particle-based MPM and the mesh-based Lagrangian approach. Chapter 3 gives more detailed derivation on the grid solve process.

- The motion of the grid is transferred back to particles to update their velocities and perform the advection step. Similar to the first step, the interpolating functions are used during the transfer, while still preserving total mass and linear momentum.

- When plastic material behavior is expected, we adopt a multiplicative decomposition of the deformation gradient $\mathbf{F} = \mathbf{F}^E\mathbf{F}^P$, where $\mathbf{F}^E$ is the elastic part and $\mathbf{F}^P$ stands for the plastic deformation. The range of admissible elastic response is defined through a function y of the stress called the yield stress, with $y \leq 0$ being the elastic region. A return mapping procedure is implemented to determine the updated $\mathbf{F}^E$ and $\mathbf{F}^P$ so that $\mathbf{F}^E$ still represents admissible material dynamics. In later chapters we provide more details on the choice of yield criteria and return mapping schemes.

Apart from the basic transfer schemes described above, various transfer methods have been developed for different purposes. In this thesis two schemes are widely used, APIC ([JSS15]) and FLIP ([BR86], [BKR88]). The details of the transfers are given where they are used in later chapters.

CHAPTER 2

Simulation and Visualization of Ductile Fracture

2.1 Introduction

Ductile materials behave elastically until a yield stress condition is met, at which point they start to deform plastically and eventually fail completely. Whether it be the distinctive patterns exhibited while tearing a piece of fruit or twisted metal after a high-velocity impact, the fracture and failure of ductile materials are ubiquitous and indispensable when creating visually interesting virtual worlds for computer graphics applications. Indeed, some of the earliest methods for simulating elasticity in computer graphics included treatment for tearing and failure of materials [TF88]. O'Brien et al. [OBH02] demonstrated that using the Finite Element Method (FEM) with continual domain remeshing after fracture events allowed for a wide range of ductile behaviors and incredibly detailed simulations. Since this pioneering approach, many others have used FEM and remeshing to achieve similar behaviors [MG04, MBF04, WTG09, WRK10]. Particle methods based on Smoothed Particle Hydrodynamics (SPH) [GBB09, CWX13] and Moving Least Squares (MLS) [PKA05, MKN04] have also been used with impressive outcome, since their unstructured nature readily allows for topological change. Procedural approaches have also achieved good results when computational cost is limited [MHH07, Cho14, JML16].

The Material Point Method (MPM) is another unstructured particle technique that naturally resolves topological changes and fracture, and also accommodates elastoplastic phenomena with ease. Furthermore, a key advantage of MPM is that the hybrid Lagrangian/Eulerian nature of the method readily resolves collisions between fragments of material. These aspects

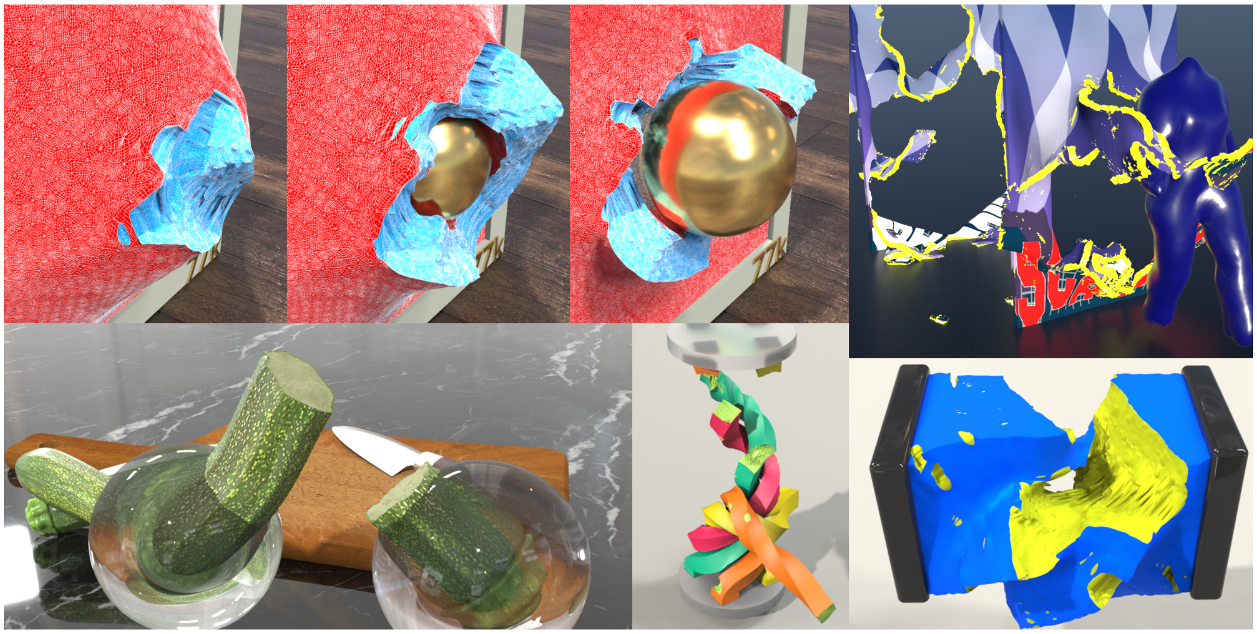


Figure 2.1: **Fracture montage**. Representative scenes from some of our simulations. **Top left**: a thin elastic sheet hit by a projectile. **Top right**: A mannequin walking through thin sheets. **Bottom left**: breaking a zucchini. Bottom middle: twisting and breaking four columns. **Bottom right**: twisting a cube.

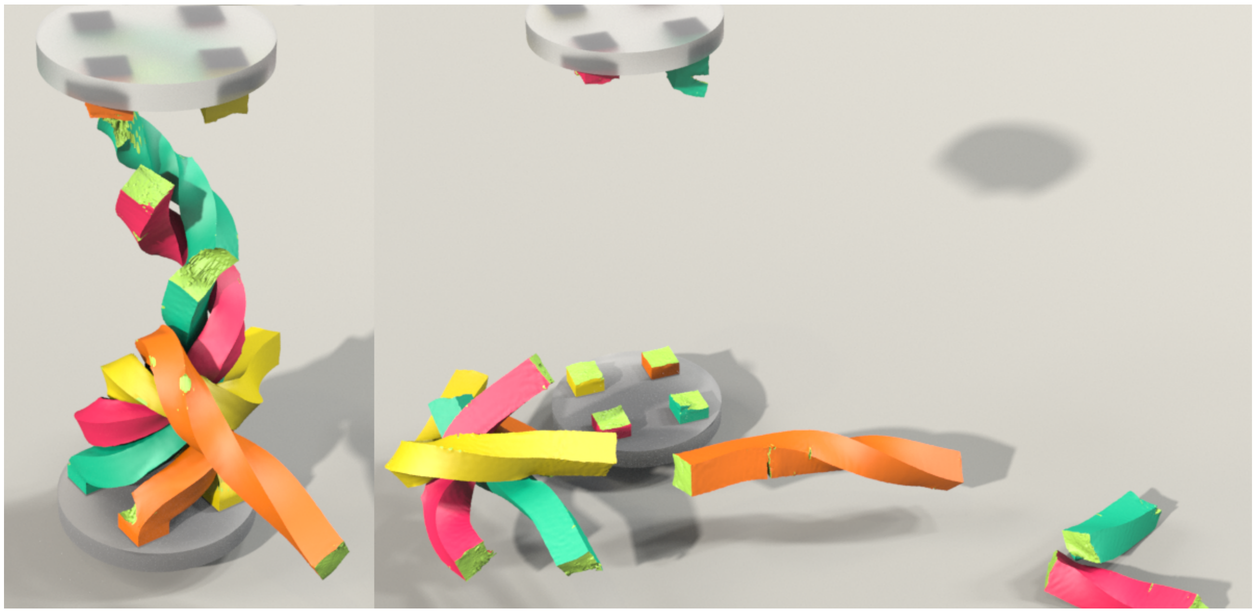


Figure 2.2: **Braided columns**. The braid is twisted and tightened until fracture occurs.

make MPM an ideal candidate for simulating fracture and failure of ductile materials. However, while MPM naturally allows for topological changes, they can be difficult to control. Particles are connected in the domain when they are in the support of the same Eulerian grid node interpolating function. Particles that do not interact with the same grid nodes in this way are decoupled. This is advantageous in that topology change requires no special treatment; however, fracture is therefore a numerical error that is not influenced by a material property but rather by discretization-related parameters like particle sampling density and Eulerian grid resolution.

Numerical fracture can be addressed by utilizing particle resampling techniques as in [YSB15] or by using the Lagrangian energy technique of Jiang et al. [JSS15] in which a tetrahedron mesh is used to compute deformation gradients. This treatment naturally couples meshed objects with MPM-based materials, and also gives an automated treatment of self-collision between meshed objects and other materials. However, in either the resampling or Lagrangian energy approaches, an additional model must be provided to allow for fracture.

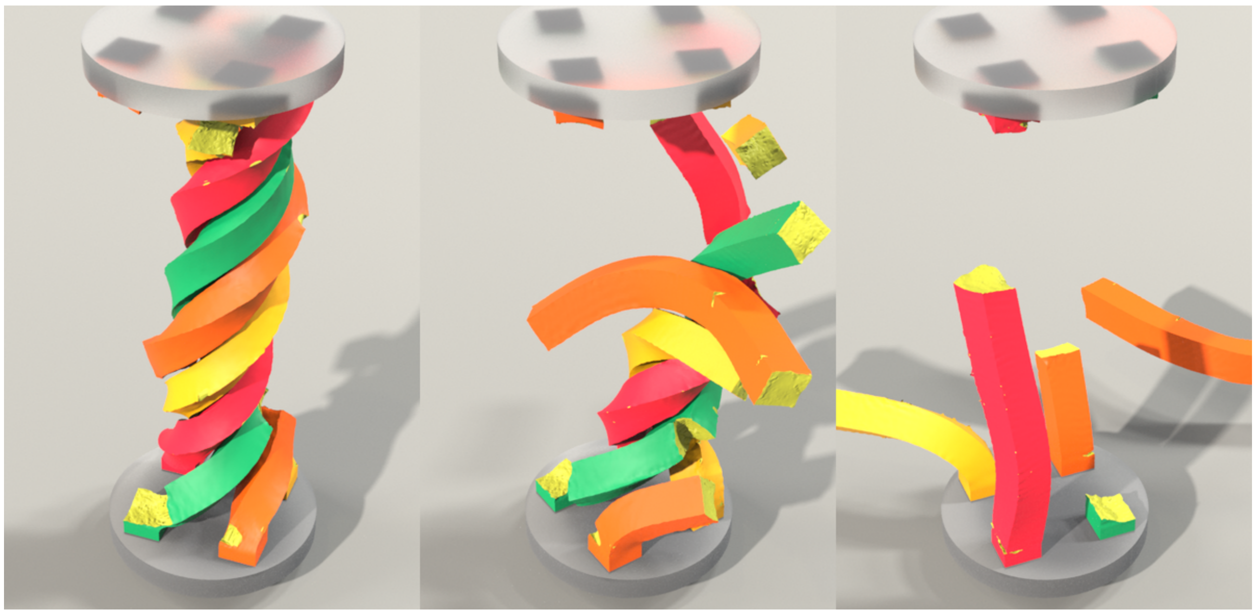


Figure 2.3: **Braided stiff columns**. Here the stiffness of the braid is increased.

A second issue hindering MPM adoption for ductile fracture is largely common to all particle-based techniques: defining and rendering material boundary surfaces in a visually sharp manner is difficult. While particle-based simulation techniques naturally allow for topological change, they generally have a more vague notion of material boundaries that complicates the process of rendering. FEM and mesh-based techniques require more intervention (remeshing) to resolve topological change, however in the process material boundaries are sharp and well defined. This is important for preserving the surface of objects created by users, and for transferring textures as the material fails.

The most common techniques for visualizing particle-based simulation data define the boundary of the particle domain as the zero isocontour of a level set function, or as a threshold value of a density function. This goes back to at least Blinn [Bli82]. Many other authors have provided improvements on these techniques over the years, including sharper surface resolution, reduction of noise and temporal coherence of surfaces, resolution of anisotropic features, and many more [MCG03, ZB05, SSP07, APK07, YT13, ATW13, MCZ07, Mus14]. However, these types of techniques are much more appropriate for fluid simulations, and

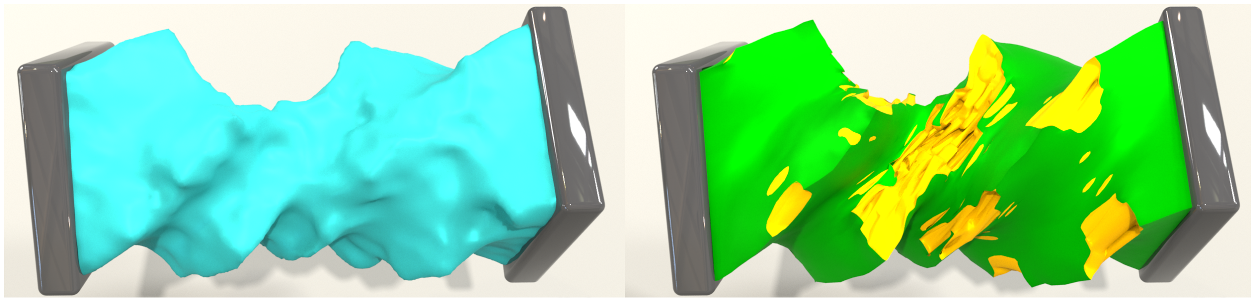


Figure 2.4: **Mesh visualization comparison**. A cube with 8000 particles is twisted to fracture in the simulation. We render the result with Houdini particle fluid surface (**left**) and our mesh visualization (**right**).

cannot support initialization from a high-resolution textured input surface mesh without complicated texture transfer at each frame, etc.

Surface tracking techniques can provide the desired preservation of sharp features and surface details. These techniques have been used with great effect in simulations of fluid [BB09, DBG14, M09, WTG10, YWT12] and viscoelastic materials [WTG09, DGP17]. These approaches are extremely powerful, but computationally expensive. However, much of the implementation and computational overhead is associated with material merging. Much simpler techniques can be used if only splitting is required. Fracture of ductile materials typically only involves failure without cohesive merging, so fully-general surface tracking techniques are not necessary.

Pre-scoring-based surfacing approaches are generally more efficient than surface tracking, and can be used when merging is not needed. These techniques predefine the maximally split configuration of the material, and only separation between components can occur. For example, the virtual node algorithm of Molino et al. [MBF04] is a pre-scoring technique where each vertex in a tetrahedron mesh represents a portion of the material in the elements in its one ring. Choi [Cho14] use a pre-scoring approach for visualizing shape-matching-based ductile fracture where each node is assigned material as a union of elements, gathered

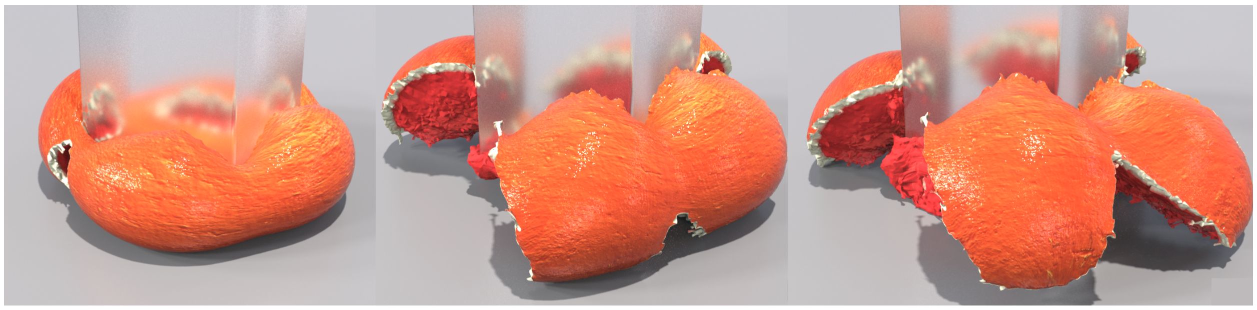


Figure 2.5: **Hydraulic press**. The orange is simulated with a meshed hollow sphere filled with guts made by MPM particles.

via K-means, from a tetrahedron mesh. Chen et al. [CZZ18] assign a single tetrahedron to each particle by initializing particles at the barycenters of an input tetrahedron mesh. In these techniques, material separation is introduced when connectivity between adjacent particle regions is severed. Crack surfaces are then defined as a subset of the boundary of the maximally split configuration. Generally, pre-scoring techniques suffer from mesh-based aliasing, since the crack paths must lie on the predefined maximally split configuration. Fracture surfaces are usually much smoother than they will appear when the sampling bias in the predefined maximally split configuration is imposed on the visualization.

We provide two options to remove the barriers preventing MPM adoption for ductile fracture simulation in graphics applications. First, we provide an extension of the mesh based strategy of Jiang et al. [JSS15] that removes numerical fracture and introduces failure through the elastoplastic constitutive equations alone. Second, when traditional particle-based MPM with numerical fracture suffices, we overcome limitations of existing surfacing strategies with a pre-scoring approach. We note that our surfacing approach is a post-process that can be implemented on data generated from standard MPM simulations. In summary, our contributions include:

- An elastoplasticity and damage model for ductile fracture that works easily with existing MPM code bases.

- A generalization of the Lagrangian energy approach of Jiang et al. [JSS15] for removing numerical fracture with ductile materials.
- A novel particle surfacing technique that preserves input surface details like texture and high-curvature regions, while removing mesh-based aliasing inherent in pre-scoring surfacing strategies.

2.2 Previous Work

Here we discuss works from the computer graphics and computational physics literature related to simulation of ductile fracture and visualization of particle-based simulation data.

Following the seminal approach of O'Brien et al. [OBH02], many authors have used FEM simulation of elastoplasticity with continual domain remeshing for ductile fracture. Müller et al. [MG04] use warped stiffness with a Rankine condition on the principal stress to define per-tetrahedron element fracture planes. Pfaff et al. [PNJ14] use an adaptive mesh to simulate tearing and cracking of thin sheets. Parker and O'Brien [PO09] use the separation tensor from [OH99] but split along element boundaries rather than cutting elements for the sake of efficiency. Wicke et al. [WRK10] dynamically remesh tetrahedron meshes to allow for efficient simulation of behaviors ranging from purely elastic to extremely plastic with fracture. Other remeshing approaches include [BWH07, WT08, WTG09, BDW13]. Wicke et al. [WBG07, KMB08] developed interpolating functions for convex polyhedral elements to allow for easy splitting of elements in fracture simulations. Gissler et al. [GBT07] introduce a notion of constraint sets for fracture simulation. Koschier et al. [KBT17] use XFEM and improve the mass matrix treatment by integrating over partially empty enriched elements. Zhang et al. [ZZS06] use tetrahedron mesh-based FEM with elastoplasticity driven damage, element splitting (at damage threshold), and molecular dynamics for debris simulation.

Pauly et al. [PKA05] use a mesh-free MLS approach to simulate elastoplastic ductile fracture

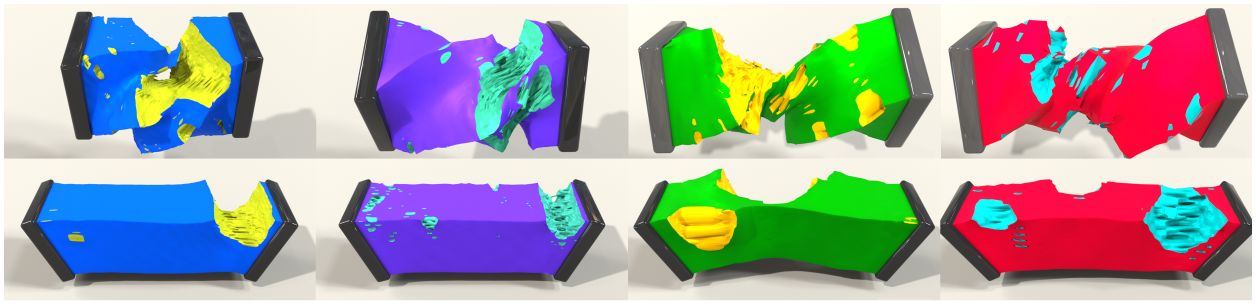


Figure 2.6: **Twist and pull**. Four identical cubes of different resolution undergoing twisting and pulling motions. From left to right: 60K, 17K, 8K, 4K particles.

with Heaviside-enriched interpolating functions, as in the XFEM approaches of Belytschko [BCX03]. They create domain and crack boundary surfaces at render time using the Surfels approach in [PKK03, WTG04]. Müller et al. [MKN04] use a similar approach. Steineman et al. [SOG09] use visibility graphs to further improve the modification of MLS interpolating functions in the presence of splitting and merging defined by explicitly tracked failure surfaces. Gerszewski et al. [GBB09] also compute the deformation gradient in a weighted least squares sense.

Other notable ductile fracture techniques include the peridynamics approach of Chen et al. [CZZ18]. Bußler et al. [BDP17] visualize crack surfaces in peridynamics particle data by computing Delaunay tetrahedralizations that respect height ridges in the damage field. Choi [Cho14] uses shape-matching to simulate procedural ductile fracture. Ohta et al. [OKN09] use an adaptive regular lattice with shape matching-based elasticity to simulate ductile fracture. Jones et al. [JML16] simulate ductile fracture using shape matching.

Various approaches for ductile fracture with MPM exist in the computational physics literature. Wretborn et al. [WAM17] simulate fracture with MPM by pre-scoring materials into pieces held together by massless particle constraints. They resolve collisions between fragments by using the MPM N-body approach of [HZM11]. Nairn et al. [Nai03, GN06] developed the CRAMP MPM technique for simulating velocity and displacement disconti-

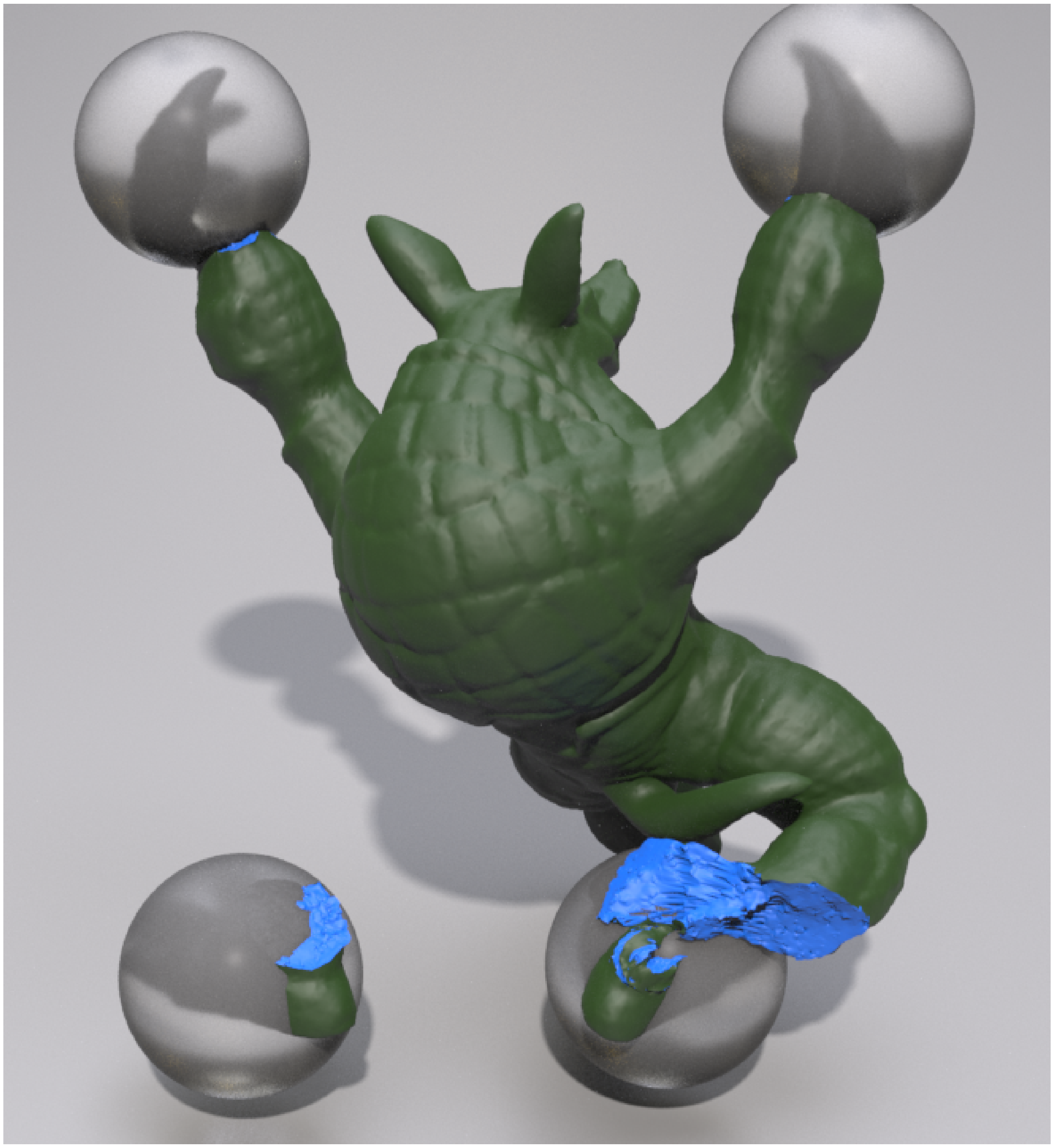


Figure 2.7: **Twist armadillo**. An armadillo twisted to fracture.

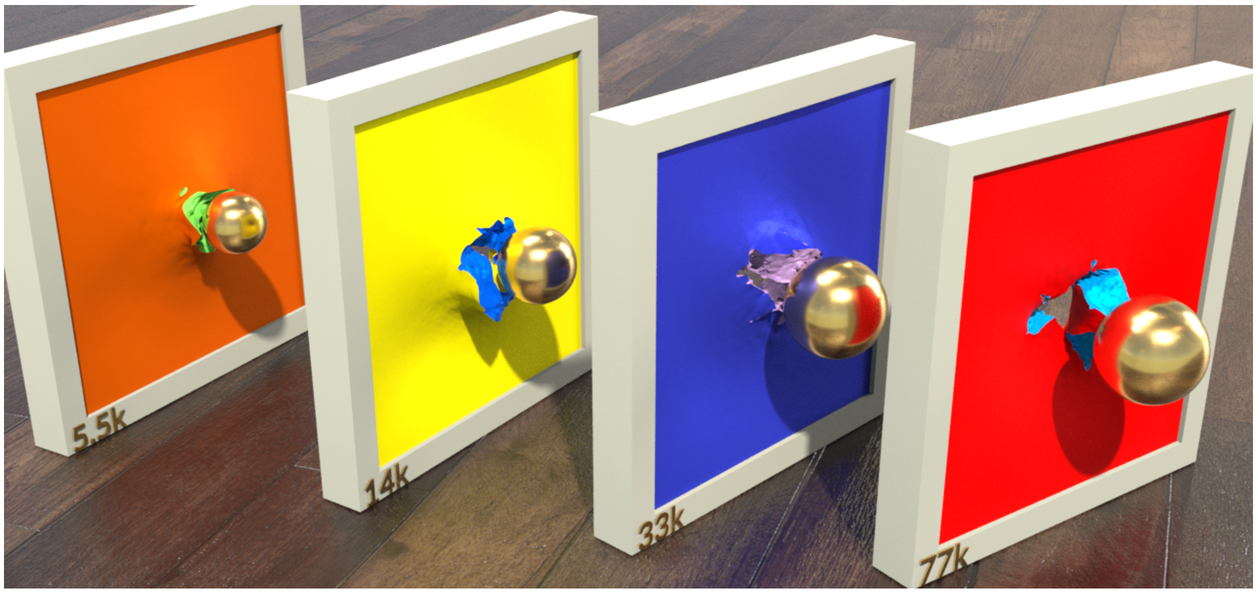


Figure 2.8: **Projectile**. Here we show shooting projectiles at ductile walls with 5.5K (orange), 14K (yellow), 33K (blue), and 77K (red) particles.

nuities on the grid. Other MPM techniques utilize grid node duplication [DLC07], and then resolve frictional contact on the duplicated Eulerian grid nodes.

Surfacing particle-based simulation data is a long-standing problem. Most approaches define the boundary of the particle domain as the zero isocontour of a level set function or as a threshold value of a density function [Bli82, DC98, MCG03, ZB05, APK07, SSP07, Mus14]. Yu and Turk developed an anisotropic approach to more accurately capture sharp features [YT13]. Bhattacharya et al. [BGB15] fit signed distance functions to particle data by minimizing a biharmonic thin shell energy over a surface constrained between interior and exterior CSG surfaces, and support anisotropic capture of sharp features as in [YT13]. Williams [Wil08] similarly solves the surfacing problem with a constrained minimization. Shen and Shah [SS07] address temporal discontinuities by blending adjacent frames. Museth et al. [MCZ07] incorporate a variety of post-processing techniques including temporal and spatial anti-aliasing. Adams et al. [APK07] use a semi-Lagrangian contouring method similar to that proposed by Bargteil et al. [BGO06]. Dagenais et al. [DGP17] improves and extends

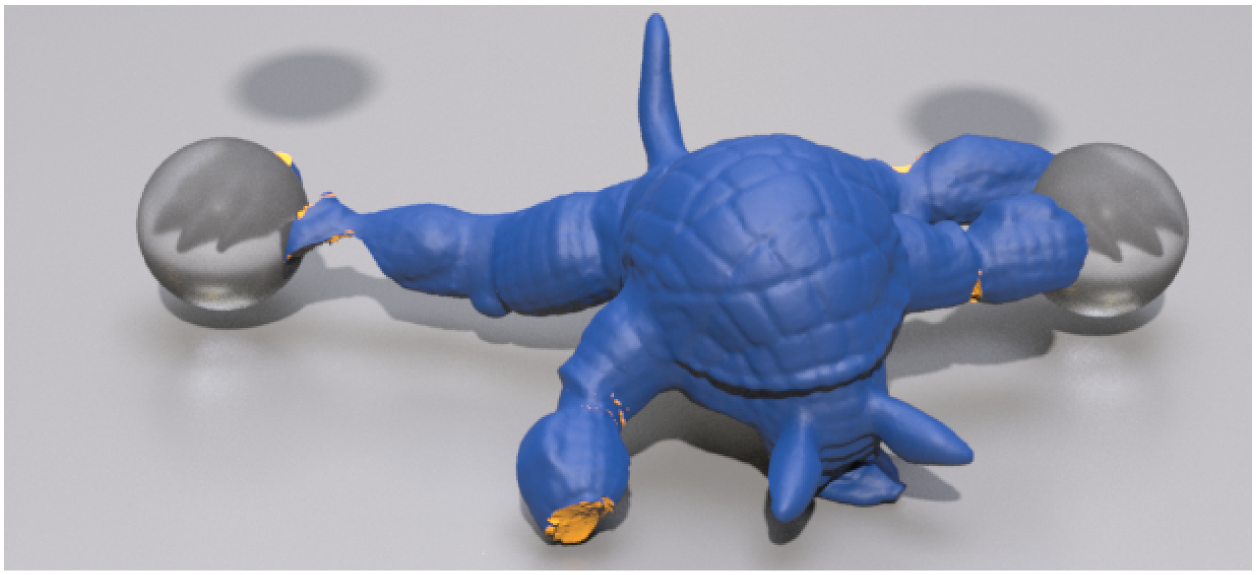


Figure 2.9: **Stretch armadillo**. An armadillo stretched to fracture.

surface tracking to retain surface details. Mercier et al. [MBT15] develop a post-process approach for surfacing particle-based fluid simulation data. They create an up-res particle surface using a generalization of the approach in [Wil08] and then apply a surface-only Lagrangian wave simulation to provide realistic, detailed motion.

Pre-scoring bodies into precomputed pieces is useful for simulation and visualization. Müller et al. [MCK13] decompose objects into convex pieces and generate fracture patterns of space using Voronoi diagrams. CSG operations are used to resolve the initial convex decomposition with the fracture patterns. Su et al. [SSF09] also fracture all of space to generate rigid body fragment pieces for real time simulation of brittle fracture. Liu et al. [LHL11] also pre-score the material along Voronoi boundaries to add user control over fracture patterns. Schvartzman and Otaduy [SO14] use Voronoi-based pre-scoring of fracture boundaries with rigid body simulation to simulate brittle fracture. Zheng and James [ZJ10] use the strain energy density to adapt Voronoi fracture regions. Raghavachary [Rag02] defines fragments in polygon meshes by splitting into Voronoi regions.

Marching cubes [LC87] is often used to create a triangle mesh for an isocontour of a scalar function fitted to the particles. Akinci et al. [AIA12] show this can be done efficiently and in parallel in a narrow band of the isocontour surface. Benber et al. [AAI12] improve this technique by applying surface decimation and subdivision algorithms. The decimation step alleviates surface bumpiness very efficiently and reduces the number of triangles in flat regions. Later, the subdivision step ensures that the non-smooth regions are smoothed. This allows for performance gains since lower resolution marching cubes grids can be used. Zhang et al. [ZLL] develop an adaptive surfacing approach using octrees and graph cuts. van der Laan et al. [LGS09] skip the marching cubes based polyhedralization for real-time efficiency and instead create surfaces based on screen-space depth. Muller et al. [MSD07] and Xiao et al. [XZY18] also work in screen space for efficiency. Rosenberg and Birdwell [RB08] optimized Marching Cubes specifically for surfacing particle surfaces.

2.3 Mathematical Background

2.3.1 Elasto-Plasticity

We use the same multiplicative decomposition as in [KGP16],

$$\mathbf{F} = \mathbf{F}^E \mathbf{F}^P \tag{2.1}$$

to capture the plastic yielding and fracturing. The plastic deformation gradient $\mathbf{F}^P$ represents the permanently damaged state of the material; no force is exerted to resist deformations represented in this part. The elastic deformation gradient $\mathbf{F}^E$, on the other hand, measures the deformation the material is still capable of defying. In the following sessions, we will describe the elastic potential energy and yield surface, which are defined in terms of $\mathbf{F}^E$ and $\mathbf{F}^P$, respectively. We will also provide the projection maps and softening rule that constitute the interaction of these two.

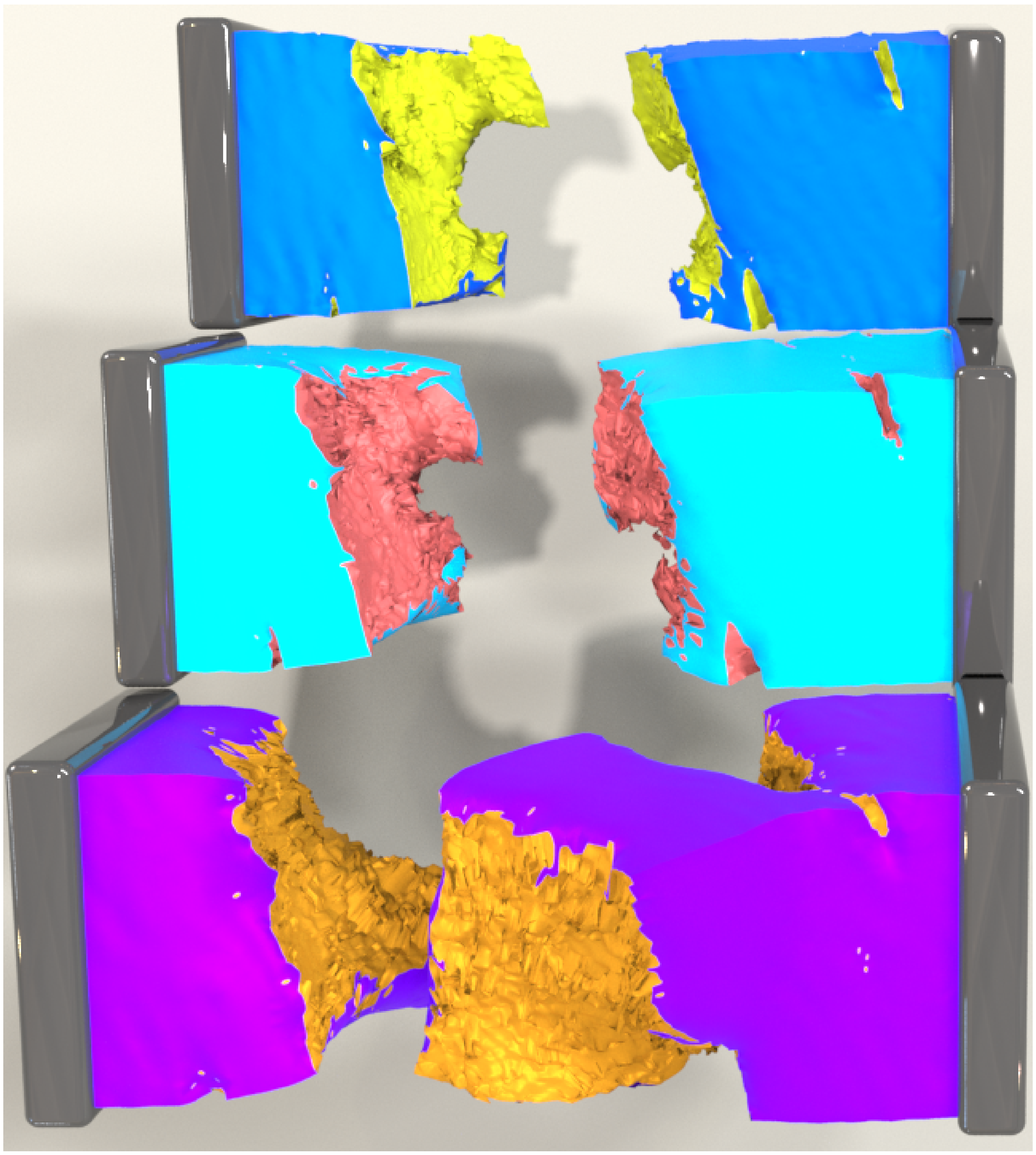


Figure 2.10: **Twist**. Twisting cubes with different von Mises yield surfaces. We use $\tau_C = E$, (blue), $0.7E$ (cyan), and $0.5E$, for Young's modulus E. Blue: $\tau_C = E$. Cyan: $\tau_C = 0.7E$. Purple: $\tau_C = 0.5E$.

2.3.2 Elastic Constitutive Model

We use the isotropic hyperelastic potential energy density of [KGP16]. This model is quadratic in elastic Hencky strain $\boldsymbol{\epsilon}^E = \frac{1}{2}\ln(\mathbf{F}^E(\mathbf{F}^E)^T)$,

$$\psi(\mathbf{F}^E) = \mu\boldsymbol{\epsilon} : \boldsymbol{\epsilon} + \frac{\lambda}{2}\mathrm{tr}^2(\boldsymbol{\epsilon}) = \mu\sum_{i=1}^{3}\ln(\sigma_i^E)^2 + \frac{\lambda}{2}\left(\sum_{i=1}^{3}\ln(\sigma_i^E)\right)^2 \tag{2.2}$$

where $\mathbf{F}^E = \mathbf{U}^E\boldsymbol{\Sigma}^E(\mathbf{V}^E)^T$ is the singular value decomposition of $\mathbf{F}^E$ and σ_i^E denote the entries in $\boldsymbol{\Sigma}^E$. Here μ and λ are the Lamé coefficients which control the amount of resistance to deformation and volume change. The Cauchy stress is defined in terms of the elastic potential as

$$\boldsymbol{\sigma} = \frac{1}{\det(\mathbf{F})}\frac{\partial\psi}{\partial\mathbf{F}^E}(\mathbf{F}^E)^T \tag{2.3}$$

$$\frac{\partial\psi}{\partial\mathbf{F}^E} = \mathbf{U}^E(\boldsymbol{\Sigma}^E)^{-1}\left(2\mu\ln(\boldsymbol{\Sigma}^E) + \lambda\ln(\boldsymbol{\Sigma})\right)(\mathbf{V}^E)^T \tag{2.4}$$

This choice of potential energy is primarily for the sake of simplifying the return mapping process (see Section 2.8.1), as discussed in [KGP16, JGT17].

2.3.3 Plasticity

Ductile materials behave elastically until a critical stress is reached, at which point deformation becomes permanent and the material achieves a new local rest state. We express this notion of critical stress in terms of a yield surface in stress space defined implicitly as $y(\boldsymbol{\sigma}) = 0$ using a yield function y. When $y(\boldsymbol{\sigma}) < 0$, the critical stress has not been achieved and the material behaves elastically. When $y(\boldsymbol{\sigma}) = 0$, the elastic limit is reached and the plastic deformation defined via $\mathbf{F}^P$ becomes non-trivial. Mathematically, we can view the dynamics of $\mathbf{F}^P$ as being chosen to satisfy the stress constraint $y(\boldsymbol{\sigma}) = 0$ through its dependence on $\mathbf{F}^E$.

Although the Cauchy stress $\boldsymbol{\sigma}$ is more physically intuitive, the Kirchhoff stress $\boldsymbol{\tau} = \det(\mathbf{F})\boldsymbol{\sigma}$ is often more convenient when working with plasticity. It is particularly convenient for defining the plastic deformation in a manner that is consistent with the second law of thermodynamics and when enforcing the yield condition discretely during time stepping, a process

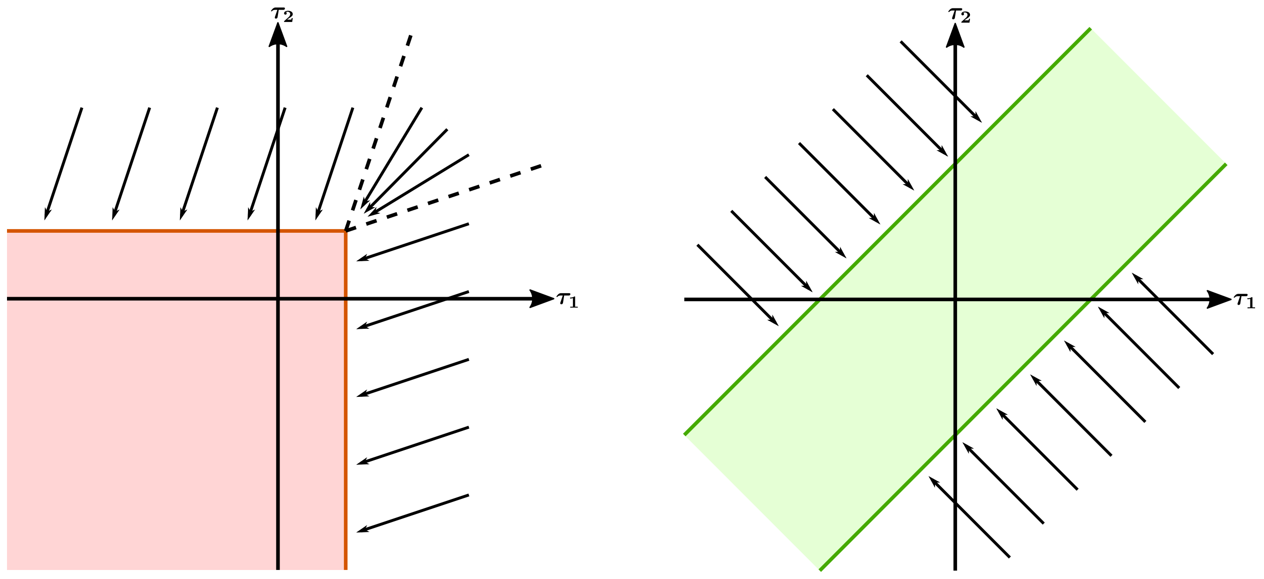


Figure 2.11: **Return mappings**. **Left**: Rankine yield surface and its return mapping. **Right**: von Mises yield surface and its return mapping.

which is typically referred to as the return mapping (see Section 2.8.1 for details). Henceforth, we will assume the yield surface is defined in terms of the Kirchhoff stress $y(\boldsymbol{\tau})$.

2.3.3.1 Yield Surface

We use two different yield surfaces to model different fracture modes. The Rankine yield surface [And17] is given by

$$y(\boldsymbol{\tau}) = \max_{\|\mathbf{u}\|=\|\mathbf{v}\|=1} \mathbf{u}^T \boldsymbol{\tau} \mathbf{v} - \tau_C \leq 0, \tag{2.5}$$

where τ_C is a scalar parameter that represents the maximum allowed tensile strength, since the expression $\max_{\|\mathbf{u}\|=\|\mathbf{v}\|=1} \mathbf{u}^T \boldsymbol{\tau} \mathbf{v}$ measures the tensile stress among all directions and corresponds to the largest eigenvalue of $\boldsymbol{\tau}$. Constraining the maximal tension in all directions enables the material to go through mode I yielding, where permanent deformation is induced in response to local tension.

The von Mises yield surface given by

$$y(\boldsymbol{\tau}) = \|\boldsymbol{\tau} - \mathrm{tr}(\boldsymbol{\tau})\mathbf{I}\|_F - \tau_C \leq 0 \tag{2.6}$$

provides plastic response to mode II and mode III shearing deformations by constraining the deviatoric (shear) stress; here $\|\mathbf{A}\|_F = \sqrt{\mathbf{A} : \mathbf{A}}$ denotes the Frobenius norm. By combining the two yield surfaces or using them independently, we can simulate a wide range of fracturing and plastic materials.

In practice, the yield condition $y(\boldsymbol{\tau}) \leq 0$ is enforced per time step. In this process, the trial strain ($\tilde{\boldsymbol{\epsilon}}^E$) is mapped from a state whose corresponding stress violates the condition to one whose corresponding stress is on the boundary of the yield surface ($\boldsymbol{\epsilon}^{E,n+1}$) in a process referred to as the return mapping. We illustrate the different yield surfaces and the associative direction for return mappings in Figure 2.11. We provide detailed derivation in Section 2.8.1.

2.3.3.2 Softening and Damage

As the material undergoes plastic deformation, we decrease τ_C to shrink the yield surface towards the origin. This limits the strength of the material as smaller and smaller stresses are admissible. For each projection $\tilde{\boldsymbol{\epsilon}}^E \to \boldsymbol{\epsilon}^{E,n+1}$ in the return mapping, we decrease τ_C by $\theta\|\boldsymbol{\epsilon} - \mathrm{proj}(\boldsymbol{\epsilon})\|_F$, where $\theta > 0$ is a material constant that defines the rate of softening.When τ_C reaches zero, we model the material as completely damaged and set the Lamé coefficients to zero.

2.4 Numerical Method

We use MPM to discretize the governing equations and cover both standard particle-based MPM as in [SCS94, SSC13] as well as the mesh-based Lagrangian energy techniques used to prevent numerical fracture [JSS15]. In the Lagrangian energy case, we modify the approach of Jiang et al. [JSS15] to include the effects of plasticity and damage.

In MPM, the discrete state consists of a collection of particles that partition the domain based on initial volumes V_p^0, with time t^n positions $\mathbf{x}_p^n$ and with masses m_p computed from the initial mass density as $\rho(\mathbf{x}_p^0, t^0)V_p^0$ and linear and affine time velocities $\mathbf{v}_p^n$, $\mathbf{C}_p^n$ used for APIC particle/grid transfers [JSS15]. In the case of traditional particle-based MPM, each particle additionally stores the elastic portion of the deformation gradient $\mathbf{F}_p^{E,n}$ and yield surface size τ_{Cp}. In the case of mesh-based MPM, we assume there additionally exists a tetrahedron mesh connecting the particles $\mathbf{x}_p^n$. We use e to denote elements in the mesh and store $\mathbf{F}_e^{E,n}$ and $\tau_{\mathbf{C}_E}$ per tetrahedron element, rather than per particle. Furthermore, in the mesh-based case, we must also store the plastic part of the deformation gradient $\mathbf{F}_e^{P,n}$.

During each MPM time step, material states are transferred from particles to grid nodes, on which the governing equations are solved numerically. The discretization is done differently in the cases of standard particle-based MPM versus the mesh-based approach. The difference lies in how the deformation gradient is computed. In the case of standard particle-based MPM, the deformation gradient is stored per particle and is updated using an updated Lagrangian view. With this assumption the deformation gradient is computed as the product of the time t^n deformation gradient $\mathbf{F}_p^n$ and the deformation of the grid (evaluated at the particle) over the time step $\hat{\mathbf{F}}_p^{n+1} = (\mathbf{I} + \Delta t \sum_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^{n+1} \nabla w_{\mathbf{i}p}^n)$ where $\nabla w_{\mathbf{i}p}^n = \frac{\partial N}{\partial \mathbf{x}}(\mathbf{x}_p^n - \mathbf{x}_{\mathbf{i}})$ is the derivative of the grid interpolating functions.

In the case of mesh-based elasticity, the deformation gradient is computed using mesh connectivity as in standard FEM [SB12, JSS15] $\mathbf{F}_e^{n+1} = \sum_p \mathbf{x}_p^{n+1} \nabla \tilde{N}_p(\mathbf{X}_e)$ where $\tilde{N}_p(\mathbf{X})$ is the piecewise linear interpolating function associated with particle p evaluated at the tetrahedron

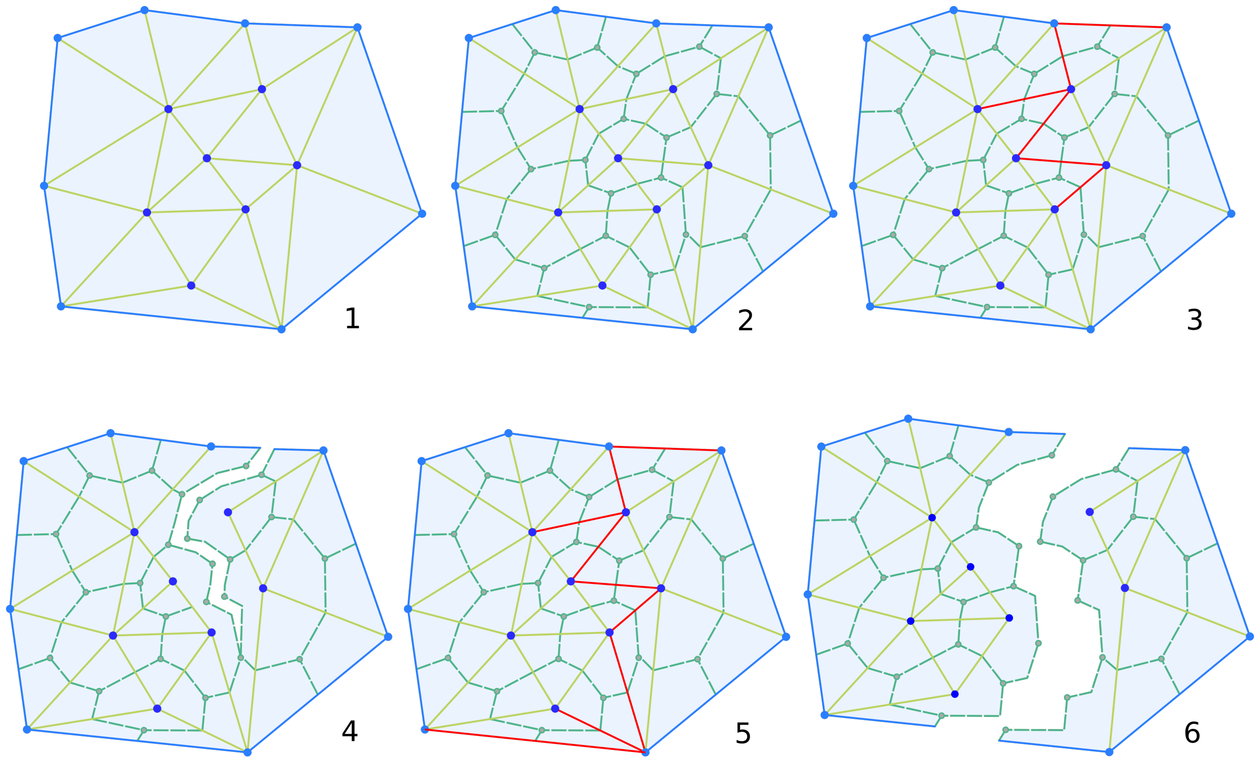


Figure 2.12: **Mesh cutting**. 1: Initial simplex mesh (Delaunay or quality mesh generated for Lagrangian simulation). 2: Particle core partitioning. 3: Identify failed edges (marked red). 4: The corresponding partially split mesh to the set of failed edges in 3. 5: A different set of failed edges (marked red). 6: The corresponding split mesh to the set of failed edges in 5.

barycenter in the initial configuration of the mesh. We summarize this below as

$$m_{\mathbf{i}}^n = \sum_p w_{\mathbf{i}p}^n m_p \tag{2.7}$$

$$\mathbf{v}_{\mathbf{i}}^n = \frac{1}{m_{\mathbf{i}}^n} \sum_p w_{\mathbf{i}p}^n m_p (\mathbf{v}_p^n + \mathbf{C}_p^n (\mathbf{x}_{\mathbf{i}} - \mathbf{x}_p^n)) \tag{2.8}$$

$$\mathbf{v}_{\mathbf{i}}^{n+1} = \mathbf{v}_{\mathbf{i}}^n + \frac{\Delta t}{m_{\mathbf{i}}^n} \mathbf{f}_{\mathbf{i}} + \Delta t \mathbf{g} \tag{2.9}$$

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \Delta t \sum_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^{n+1} w_{\mathbf{i}p}^n \tag{2.10}$$

$$\mathbf{v}_p^{n+1} = \sum_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^{n+1} w_{\mathbf{i}p}^n \tag{2.11}$$

$$\tilde{\mathbf{C}}_p^{n+1} = \frac{12}{\Delta x^2 (b+1)} \sum_{\mathbf{i}} w_{\mathbf{i}p}^n \mathbf{v}_{\mathbf{i}}^{n+1} \otimes (\mathbf{x}_{\mathbf{i}} - \mathbf{x}_p^n) \tag{2.12}$$

$$\mathbf{C}_p^{n+1} = (1-\nu)\, \tilde{\mathbf{C}}_p^{n+1} + \frac{\nu}{2} \left(\tilde{\mathbf{C}}_p^{n+1} - (\tilde{\mathbf{C}}_p^{n+1})^T \right) \tag{2.13}$$

$$\tilde{\mathbf{F}}_e^E = \left(\sum_p \mathbf{x}_p^{n+1} \nabla \tilde{N}_p (\mathbf{X}_e) \right) (\mathbf{F}_e^{P,n})^{-1} \tag{2.14}$$

$$\tilde{\mathbf{F}}_p^E = (\mathbf{I} + \Delta t \sum_{\mathbf{i}} \mathbf{v}_{\mathbf{i}}^{n+1} \nabla w_{\mathbf{i}p}^n) \mathbf{F}_p^{E,n} \tag{2.15}$$

$$\mathbf{F}_q^{E,n+1} = \text{returnMap}(\tilde{\mathbf{F}}_q^E). \tag{2.16}$$

Here the particle-to-grid transfers consist of Equations (2.7)-(2.8); the grid-based momentum update step consists of Equations (2.9)-(2.11); and the interpolation from grid to back particles is implemented with Equations (2.11)-(2.13). We use APIC transfers [JSS15] for Equations (2.8) and (2.12) as well as RPIC damping of [JGT17] in Equation (2.13) where ν controls the amount of damping. In Equation (2.9), $\alpha = 0$ corresponds to a symplectic Euler update and $\alpha = 1$ corresponds to backward Euler. Equations (2.14) and (2.15) represent the deformation gradient update in the cases of mesh-based and standard MPM respectively. Equation (2.16) projects the elastic state to satisfy the plasticity constraints. The equation is indexed by q to indicate that it is either e for mesh-based or p for particle-based MPM.

In Equation (2.9), $\mathbf{f}_{\mathbf{i}}$ is the force on grid node $\mathbf{i}$ which is computed as the variation of the total potential with respect to grid nodes moving as $\mathbf{x}_{\mathbf{i}} + \Delta t \mathbf{v}_{\mathbf{i}}^{n+\alpha}$, where $\alpha = 0$ corresponds to symplectic Euler and $\alpha = 1$ corresponds to backward Euler time stepping. The value varies based on the choice of mesh- or particle-based MPM as

$$\mathbf{f}_\mathbf{i} = \begin{cases} \sum_p w_{\mathbf{i}p}^n \mathbf{f}_p(\mathbf{x}^{n+\alpha}) + \Delta t \mathbf{g}, \\ -\sum_p \frac{\partial \psi}{\partial \mathbf{F}^E}(\tilde{\mathbf{F}}_p^E(\tilde{x}^{n+\alpha}))(\mathbf{F}_p^{E,n})^T \nabla w_{\mathbf{i}p}^n V_P^0 + \Delta t \mathbf{g} \end{cases} \tag{2.17}$$

respectively, where $\mathbf{x}^{n+\alpha} \in \mathbb{R}^{2n_P}$ is the vector consisting of all particle time $t^{n+\alpha}$ positions $\mathbf{x}_p^{n+\alpha}$ according to Equation (2.10). In the case of standard particle MPM, $\tilde{\mathbf{x}}^{n+\alpha}$ is the vector of all Eulerian grid node positions, moved according to

$$\mathbf{x}_\mathbf{i}^{n+\alpha} = \begin{cases} \mathbf{x}_\mathbf{i}, & \alpha = 0 \\ \mathbf{x}_\mathbf{i} + \Delta t \mathbf{v}_\mathbf{i}^{n+1}, & \alpha = 1 \end{cases} \tag{2.18}$$

In the case of mesh-based MPM, the particle force $\mathbf{f}_p$ in Equation (2.17) is related to the variation of the potential as estimated over the tetrahedron mesh, rather than the particles

$$\mathbf{f}_p = \sum_e \frac{\partial \psi}{\partial \mathbf{F}^E}(\tilde{\mathbf{F}}_e^E(\mathbf{x}^{n+\alpha})) \nabla \tilde{N}(\mathbf{X}_e) \tag{2.19}$$

where $\tilde{\mathbf{F}}_e^E(\mathbf{x}^{n+\alpha})$ is given by Equation (2.15).

2.5 Material Surface Definition and Visualization

We provide a novel pre-scoring strategy for visualization of material boundary and crack surfaces as a post-process for ductile fracture simulations. Our approach can easily be used for most standalone MPM solvers. Our technique works with either traditional particle-based MPM, or Lagrangian energy mesh-based MPM [JSS15]. In the case of mesh-based MPM, we assume the user provides a tetrahedron mesh of quality suitable for FEM simulation of elasticity. In the case of traditional particle-based MPM, we assume the user provides interior points that are sampled with a Poisson disc, or similar initial random spacing. We also assume that the user provides a triangulation of the boundary of the domain from which the internal particles are sampled. The vertices of the boundary (triangle) mesh and the

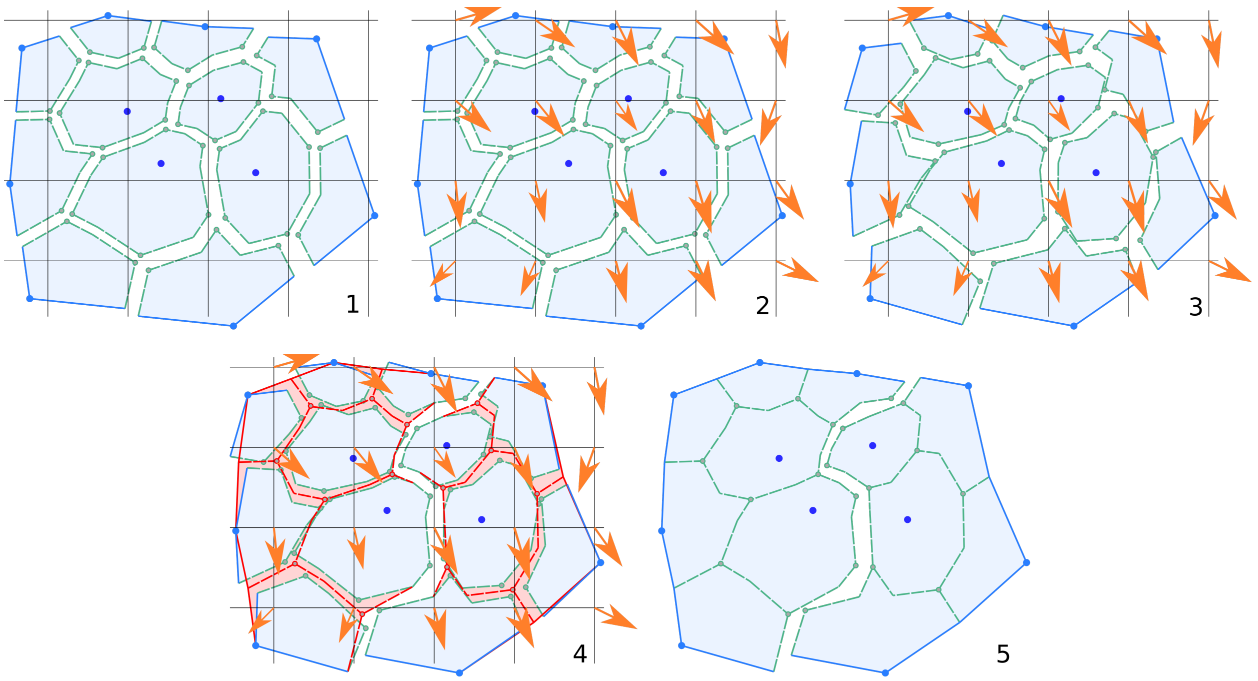


Figure 2.13: **Extrapolation**. 1. Initial particle core partition. 2. Velocity field defined on grid. 3. Particle cores positioned and oriented by local rigid body transform. 4. Sewing connected cells. 5. Final deformed fractured mesh.

randomly sampled interior particles are treated as MPM particles for simulation. If the user does not provide a triangle mesh, we can generate one by surfacing the interior particles using an existing technique like [YT13]. We assume that most users will define the boundary of the initial domain for ductile materials using a triangle mesh, typically with texture etc. and our approach is designed to preserve those details throughout the simulation. Once in possession of the boundary triangle mesh and the interior particles, we create a Delaunay tetrahedralization connecting the interior and boundary points and preserving triangles on the original boundary.

2.5.1 Mesh Topology Visualization

With our initialization strategy, in either the traditional particle-based MPM or Lagrangian energy mesh-based MPM cases, we can assume we have a tetrahedralization of the particles used in the MPM calculation. The mesh is used to define a particle-wise partition of the material domain. Each tetrahedron in the mesh is split into four cuboids, one for each of its particles. To create the particle-wise partitioning, each particle in the MPM calculation receives a cuboid from each of the tetrahedron elements it belongs to. We note that this is essentially the same as the per-particle cores of material used in the virtual node approach of Molino et al. [MBF04]. We adopt this name and refer to the particle's union of cuboids as its core of the domain. With this convention, each particle is responsible for updating its core over the course of the simulation.

The boundary of each particle core initially shares faces with cores of particles that it is connected to in the tetrahedron mesh. We define material failure on a per-initial-tetrahedron-mesh-edge basis. That is, common faces on cores of material associated with particles initially connected in the tetrahedron mesh are treated as identical until material failure occurs. To define material failure, we label core faces between particles connected along an edge in the tetrahedron mesh as broken. We use a simple union-find data structure to manage the topological connectivity and create a hexahedron mesh that respects the failed core faces. To do this we start with a mesh that is completely broken into the maximally split configuration and merge unbroken faces using the union-find data structure. See Figure 2.12 for details. One could use an element wise splitting strategy where core faces within a damaged element are broken, but we found that this gave inferior results to this edge-wise criterion.

We manage all topological aspects of the material and crack surface visualization with this simple strategy. Next we discuss our criteria for deciding when an edge (and its associated core faces) are broken as well as the geometric aspects of the crack surface evolution.

2.5.2 Topology Evolution

We use a history-based maximal stretching criteria to define broken edges. We define the maximum relative stretching of an edge for times before a given time t as

$$\zeta_t = \max_{s<t} \frac{\|\phi(\mathbf{X}_1, t) - \phi(\mathbf{X}_2, t)\|}{\|\mathbf{X}_1 - \mathbf{X}_2\|}. \tag{2.20}$$

When this value is larger than a threshold, we consider the cores associated with $\mathbf{X}_1$ and $\mathbf{X}_2$ as separated from each other and break the edge connecting them. Note that if any edge is broken at a given time $\hat{t}$ it will be broken for all times $t > \hat{t}$.

2.5.3 Mesh Geometry Visualization: Extrapolation

Each particle is responsible for updating the configuration of its core. We do this with a simple extrapolation strategy. We use a rigid transform local to each particle to extrapolate the motion of the particle to the rest of its core. For each core vertex $\mathbf{y}_p^n$ associated with a particle center $\mathbf{x}_p^n$, we compute the time t^n position as

$$\mathbf{y}_p^n = \mathbf{R}_p^n(\mathbf{y}_p^0 - \mathbf{x}_p^0) + \mathbf{x}_p^n, \tag{2.21}$$

where $\mathbf{R}_p^n$ is the rotation associated with the simulated particle p at time t^n. See Figure 2.13 for details.

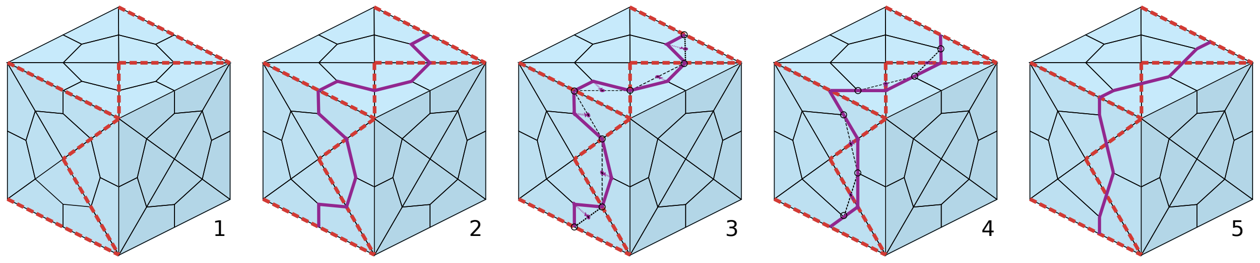


Figure 2.14: **Crack boundary curve smoothing**. From left to right: 1: Identify broken edges (red dashed line). 2: Identify boundary curve of the crack surface (purple solid line). 3-4: Smooth crack boundary curve while remaining on the original boundary surface: triangle centers move to average of neighbors, edge centers move to the intersection of its associated edge and the path joined by its neighbors. 5: Crack boundary curve after one iteration of smoothing.

We use the MPM grid velocity to update the local rotation matrix on each particle

$$\mathbf{Z}_p^{n+1} = \left(\mathbf{I} + \sum_i \tilde{\mathbf{v}}_i^n \nabla \omega_{ip}^n\right) \mathbf{R}_p^n, \tag{2.22}$$

$$\mathbf{R}_p^{n+1}\mathbf{S}_p^{n+1} = \mathbf{Z}_p^{n+1}. \tag{2.23}$$

where the polar decomposition $(\mathbf{R}_p^{n+1})^T\mathbf{R}_p^{n+1} = \mathbf{I}$, $\mathbf{S}_p^{n+1} = (\mathbf{S}_p^{n+1})^T$ is used to enforce orthogonality. This creates a rigid core translating and rotating with the particle. However, when the vertices on the boundary of the core are associated with multiple cores, we take the average of the extrapolated positions given by each core. This introduces visually realistic deformation when material is not fully failed, while reverting to translation and rotation in the event of a fully separated core.

The accuracy of the update in Equation (2.22) is affected by the particle sampling density. If the grid resolution is too high relative to the particle density, the update can be noisy. For traditional particle-based MPM this is not an issue, however for Lagrangian energy MPM we found it advantageous to add traditional MPM particles in each element to help resolve update in Equation (2.22). These particles are not used to compute forces until

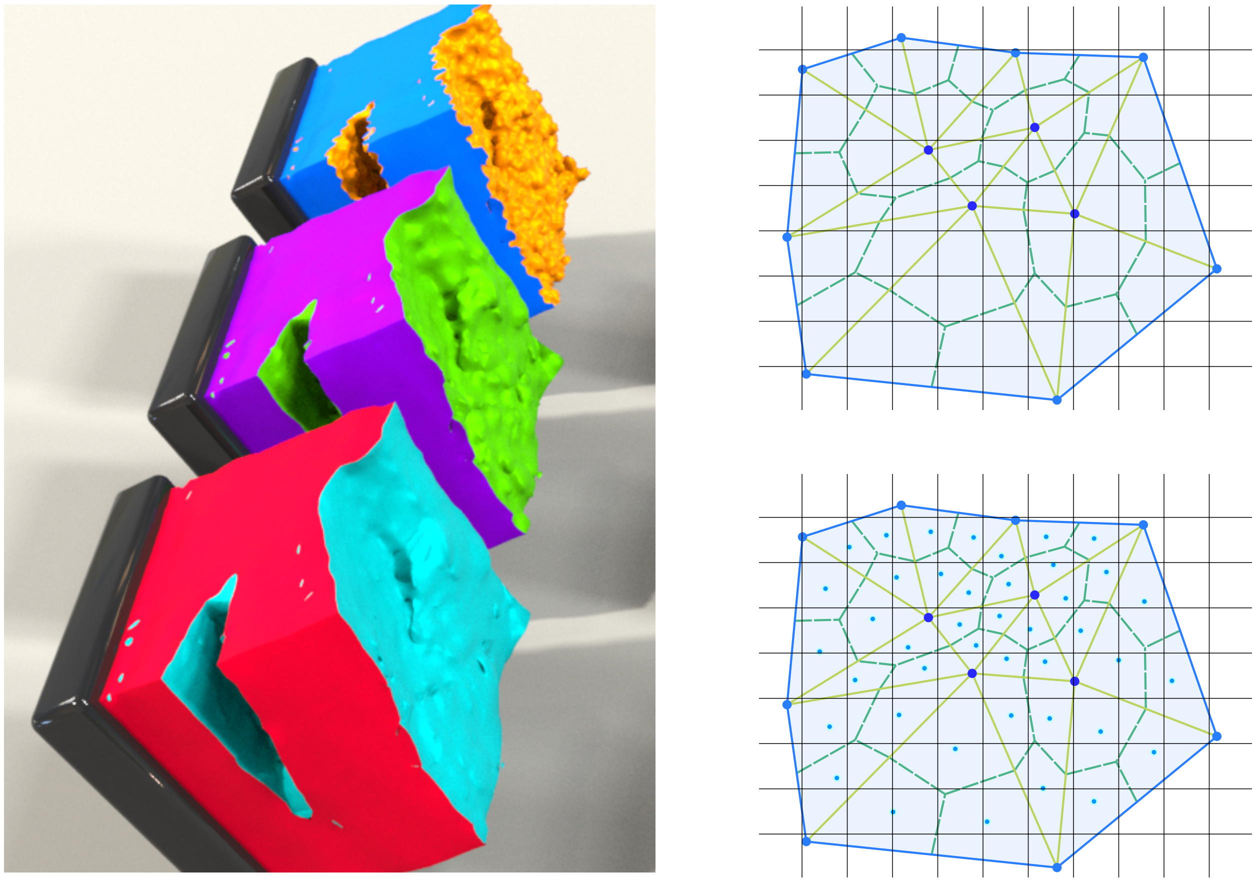


Figure 2.15: **Smoothing comparison**. **Left**: original crack surface (yellow), crack surface smoothed with 2 iterations (green), crack surface smoothed with 20 iterations (cyan). **Right**: we sample extra particles in each quadrilateral/cuboid to help reduce noise.

their parent elements fail. In the event of failure, they function as standard elastic MPM particles. See Figure 2.15 on the right for details.

2.5.4 Mesh Geometry Visualization: Crack Smoothing

There is considerable flexibility when defining the initial geometry of each particle core. The geometry of the cuboid is most naturally chosen by setting its vertices as the edge, face and tetrahedron centers. However, these points may be chosen anywhere in their respective submanifolds. The only points on the cuboids without flexibility are those corresponding to MPM particles (tetrahedron mesh vertices). We take advantage of this flexibility to remove sampling based biasing in the crack paths. Note that the flexibility is only in the initial geometry of the cuboids. Once set, they must always evolve according to the per-particle extrapolation in Section 2.5.3.

A limitation of our pre-scoring visualization approach is that all possible crack paths are determined from the initial particle partitioning of the domain. This will lead to sampling bias of the crack surface in general. This tends to make the crack surfaces appear more jaggy when initial points are randomly sampled; while the structure is imposed on the crack paths in the case of structured initial points. In order to remove initial sampling bias, we iteratively smooth the crack surface in the initial configuration. Smoothing the surface tends to remove sampling bias as is usually visible through regions of locally high curvature. Since our visualization technique is a post-process, we can assume that we know the topology of the crack surface at the final time from the condition in Section 2.5.2. We can therefore smooth the entire surface in the initial configuration, as required.

The first step of our approach involves smoothing the intersection of the initial material boundary surface and the crack surface. Care must be taken in this step to ensure that the boundary crack curves remain on the initial boundary during the smoothing process. See Figure 2.14 for details. Next, we smooth the crack surface interior by assigning each vertex to the average of its neighbors while the curve processed in the first step remains unchanged. We do this in a Gauss-Seidel fashion. Our approach quickly removes high-frequency noise while preserving the general shape of the crack pattern.

2.6 Results

We demonstrate our ductile fracture simulation and surface visualization techniques with a variety of simulations exhibiting a wide range of representative behaviors. We list our

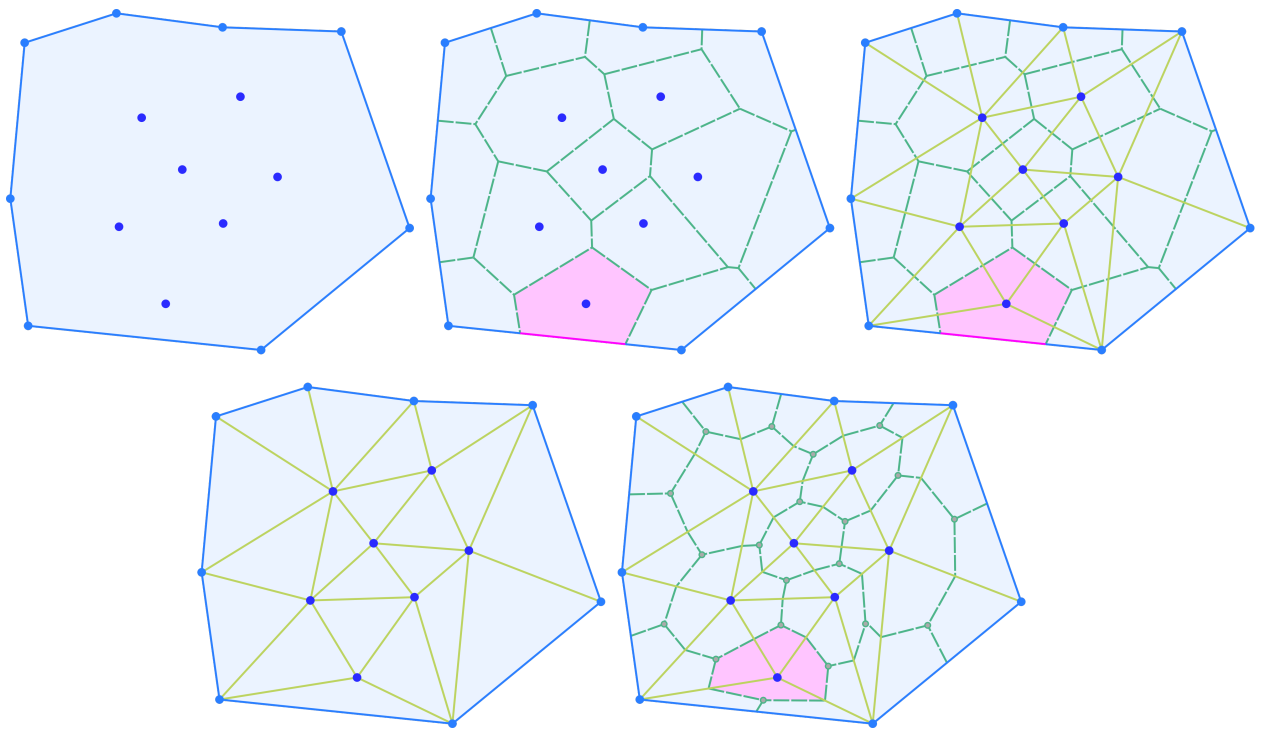


Figure 2.16: **Voronoi versus Delaunay**. Given a point cloud and a boundary surface, its Voronoi diagram could be ill-posed where interior cell intersects the boundary. If we take the dual of the Voronoi diagram, its Delaunay triangulation, then we can construct the degenerated Voronoi region without interior cell contacting the boundary.

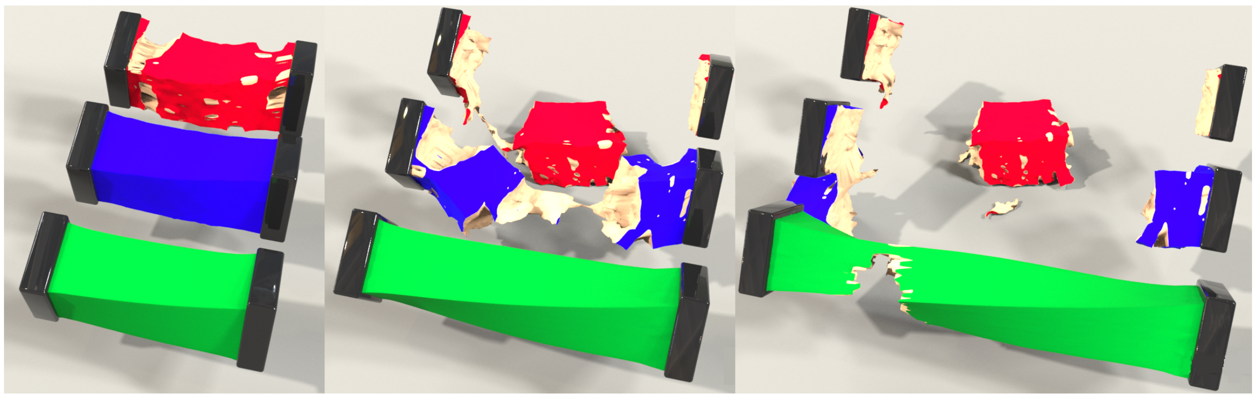


Figure 2.17: **Pull comparison**. We illustrate our treatment of numerical fracture with three simulations using the same particles. The red cube and blue cubes are simulated using traditional particle-based MPM with fine grid resolution (approximately 1 particle per grid cell) and coarse grid resolution (approximately 6 particles per grid cell) respectively. The green cube is simulated with our Lagrangian approach and fine grid resolution. Red: particle MPM with fine grid. Blue: particle MPM with coarse grid, Green: Lagrangian MPM with fine grid. Lagrangian isn't affected by numerical fracture.

computational performance and simulation details in Table 3.1. We note that in many of our examples, remarkably detailed fracture patterns are produced with comparatively low resolutions. This is advantageous because surfacing limitations often require simulations with artificially high resolution in many MPM applications. Our results were run on an Intel Xeon E5-2687W v4 with 48 threads. Time stepping was adaptively chosen according to the CFL condition, i.e. Δt was set so no particle travels more than a portion of a grid cell in each time step. For particle-based MPM, the grid resolution was chosen so that there are initially approximately six particles per grid cell. For Lagrangian energy MPM, the grid resolution reflects the tetrahedron mesh resolution, i.e. grid Δx was chosen roughly the same as the average edge length of the tetrahedron mesh. In our examples, we used TetWild to generate the tetrahedron mesh for Lagrangian MPM [HZG18].

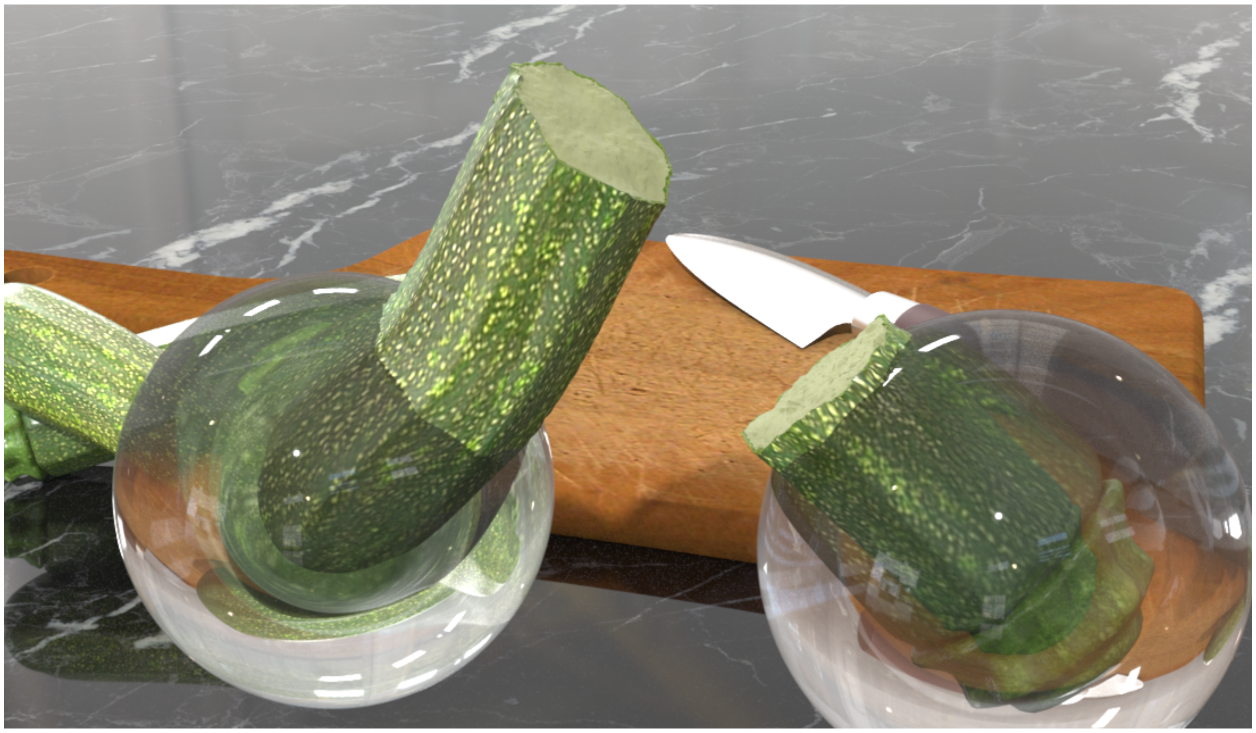


Figure 2.18: **Zucchini**. We break a zucchini in half and show the sharp crack surface.

2.6.1 Capturing Different Fracture Modes

We test our method with fracture simulations in which excessive tension or shear force is applied. In Figure 2.17, we simulate the process of pulling on a cube and demonstrate how Lagrangian MPM prevents numerical fracture caused by excessive deformation. In Figure 2.6, we twist and pull a cube until the shearing forces cause material failure and the material becomes disconnected. In Figure 2.7, we pull the 4 limbs of the armadillo until they break and observe how the fracture introduces momentum to the torso. In Figure 2.10, we added the von Mises plasticity model to the particles to capture more shear-induced plastic deformation.

2.6.2 Texturing Objects

Our mesh visualization technique has the advantage that it naturally accommodates texturing based on an input mesh. E.g. all particles from the initial mesh are in the cut mesh and

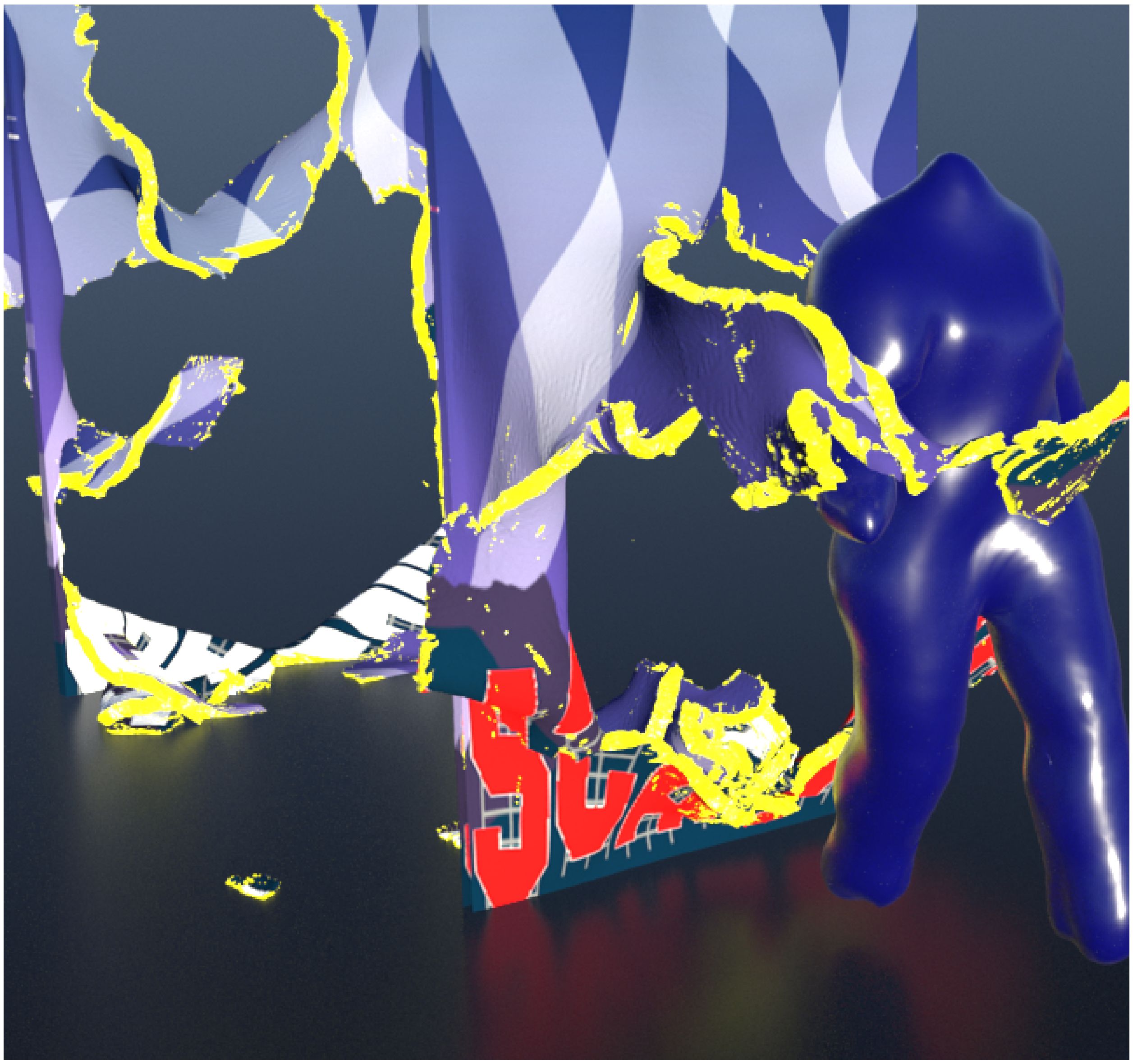


Figure 2.19: **Mannequin**. A mannequin walking through and breaking a ductile thin sheet.

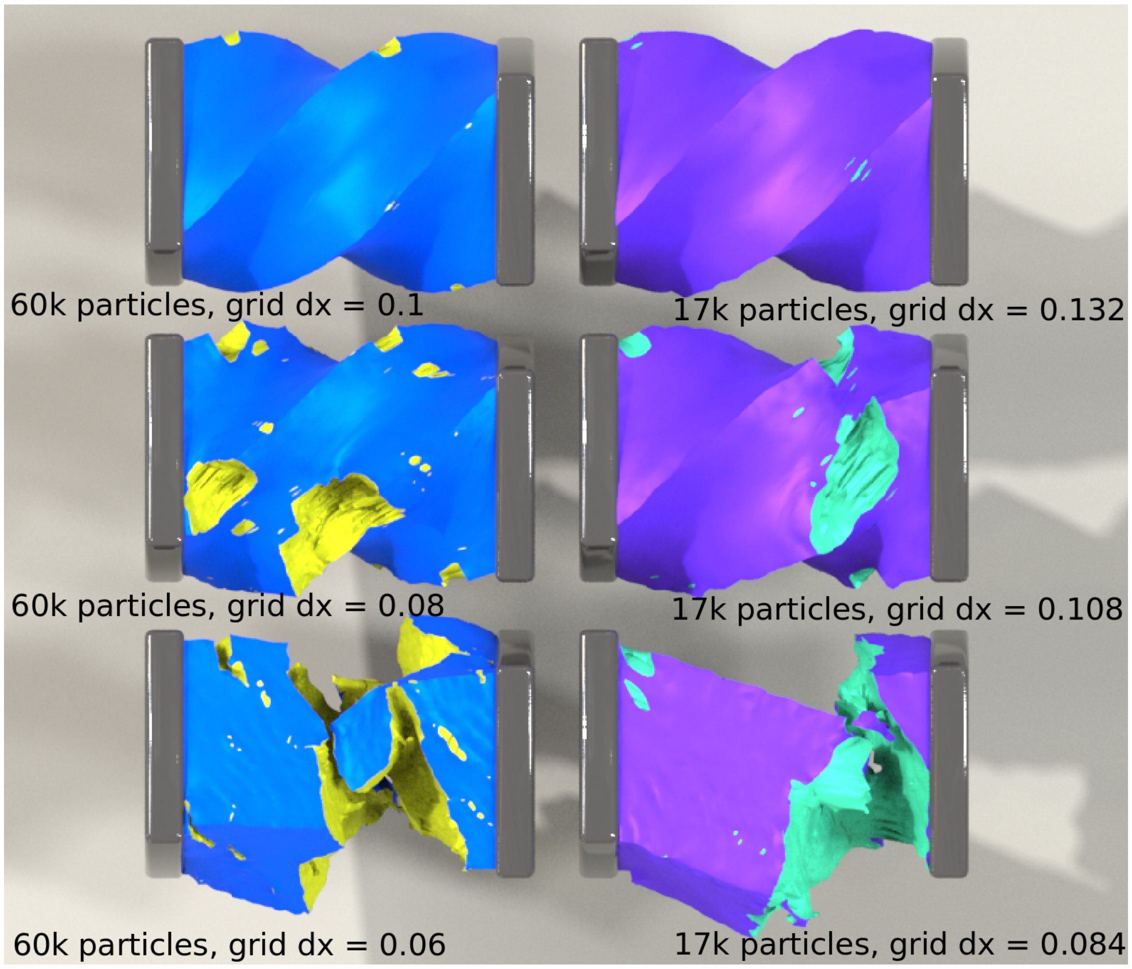

60k particles, grid dx = 0.1
17k particles, grid dx = 0.132
60k particles, grid dx = 0.08
17k particles, grid dx = 0.108
60k particles, grid dx = 0.06
17k particles, grid dx = 0.084

Figure 2.20: **Twist comparison**. We compare the same twisting cube simulation with different particle count and grid size. The sims with smaller grid dx to particle count ratio experience more fracture than the ones with larger ratio in the same frame.

it is trivial to obtain a consistent vertex ordering based on the initial mesh for simplified texturing. In Figure 2.18, we simulated a zucchini being broken in half and demonstrated that its detailed texture is preserved. In Figure 2.19, we textured a ductile thin sheet with SCA logos broken by the walking mannequin. In Figure 2.5, we textured the ductile sphere and created convincing details in the fracture scene.

2.6.3 Relaxed Resolution Requirements

In Figure 2.4, we simulated twisting of a cube with 8,000 particles. We compared two different renders: conventional particle fluid surface reconstruction and our approach. Our result captures significantly more detail and does not suffer from reconnection due to proximity. We also provide similar resolution comparison in Figure 2.6, and Figure 2.8. The rendered results still look sensible even with comparatively low resolution with our meshing technique.

2.7 Discussion and Limitations

Many existing FEM approaches for simulating ductile materials rely on the creation of a sufficiently high quality tetrahedron mesh to be used in the simulation. In the case of traditional particle based MPM, our mesh quality demands are practically non-existent. Indeed we simply use Delaunay tetrahedralization. In the case of Lagrangian mesh-based MPM our approach requires a mesh with the same quality constraints as traditional FEM. In either case, the MPM conception of our approach automatically resolves self-collision allowing us to simulate ductile fracture with comparably low implementation and computational complexity. Our approach does have a number of clear limitations. First, crack patterns are affected by particle sampling density/tetrahedron mesh topology and grid resolution, see Figure 2.20. Also, choosing appropriate parameters for edge splitting thresholds and crack surface smoothing iteration counts can vary from example to example.

Acknowledgements

The work is 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 Adobe Inc.

	Sim	Post-process	Res
Pull - MPM (Fig. 2.17 red and blue)	0.6	0.5	8K
Pull - Lagrangian (Fig. 2.17 green)	0.6	0.5	8K
Projectile - 77K (Fig. 2.8 red)	2	5	77K
Projectile - 33K (Fig. 2.8 blue)	0.9	2	33K
Projectile - 14K (Fig. 2.8 yellow)	0.4	0.7	14K
Projectile - 5.5K (Fig. 2.8 orange)	0.2	0.3	5.5K
Twist - 60K (Fig. 2.6 blue)	11	5	60K
Twist - 17K (Fig. 2.6 purple)	4	1	17K
Twist - 8K (Fig. 2.6 green)	2	0.4	8K
Twist - 4K (Fig. 2.6 red)	2	0.2	4K
Twist von Mises (Fig. **??**)	11	4	60K
Pulling with angle - 60K (Fig. 2.6 blue)	11	5	60K
Pulling with angle - 17K (Fig. 2.6 purple)	8	1	17K
Pulling with angle - 8K (Fig. 2.6 green)	8	0.4	8K
Pulling with angle - 4K (Fig. 2.6 red)	5	0.2	4K
Braiding Columns (see supplementary video)	2	3	50K
Braiding Columns (Fig. 2.2 and Fig. 2.3)	35	16	200K
Crushing Orange (Fig. 2.5)	15	8	130K
Zucchini (Fig. 2.18)	16	13	207K
Stretching Armadillo (Fig. 2.9)	49	27	299K
Tearing Armadillo (Fig. 2.7)	48	26	299K
Mannequin (Fig. 2.19)	50	5	933K

Table 2.1: Our simulations and post-processes were run with 48 threads and 128 GB of RAM. Simulation and post-process time are measured in averaged seconds per frame, and resolution is measured by particle count.

2.8 Appendix

2.8.1 Return Mapping

A trial state of deformation $\tilde{\mathbf{F}}^E$ is computed, assuming no plastic flow from time t^n to t^{n+1}. With this assumption, the plastic deformation does not change over the time step, so $\mathbf{F}^{P,n+1} = \mathbf{F}^{P,n}$, and $\mathbf{F}^{E,n+1} = \tilde{\mathbf{F}}^E$. However, if the yield condition is violated when $\boldsymbol{\tau}$ is computed from the trial deformation $\tilde{\mathbf{F}}^E$, then $\tilde{\mathbf{F}}^E$ must be modified accordingly to satisfy the constraint. This process is often referred to as the return mapping: $\tilde{\mathbf{F}}^E \to \mathbf{F}^{E,n+1}$. There are infinitely many ways that this can be done. We use associative plastic flow since it is straightforward with our choice of hyperelastic potential, and guarantees no violation of the second law of thermodynamics.

Associativity requires that the projection of the stress be done in a direction equal to the elasticity tensor $\mathcal{C} = 2\mu\mathcal{I} + \lambda\mathbf{I} \otimes \mathbf{I}$ times the normal to the yield surface $\frac{\partial y}{\partial \boldsymbol{\tau}}$. Here $\mathcal{C}$ is a fourth-order tensor, $\mathcal{I}$ the fourth-order identity tensor, and $\mathbf{I}$ the second-order identity tensor. This process can be described succinctly in terms of the trial and project elastic Hencky strain as

$$\tilde{\boldsymbol{\epsilon}}^E - \boldsymbol{\epsilon}^{E,n+1} = \delta \frac{\partial y}{\partial \boldsymbol{\tau}}(\mathcal{C} : \boldsymbol{\epsilon}^{E,n+1}), \tag{2.24}$$

where $\tilde{\boldsymbol{\epsilon}}^E = \frac{1}{2}\ln(\tilde{\mathbf{F}}^E(\tilde{\mathbf{F}}^E)^T)$ is the trial elastic Hencky strain, $\boldsymbol{\epsilon}^{E,n+1} = \frac{1}{2}\ln(\mathbf{F}^{E,n+1}(\mathbf{F}^{E,n+1})^T)$ is the projected elastic Hencky strain, $\mathcal{C} : \boldsymbol{\epsilon}^{E,n+1} = \boldsymbol{\tau} = \lambda\mathrm{tr}(\boldsymbol{\epsilon}^{E,n+1})\mathbf{I} + 2\mu\boldsymbol{\epsilon}^{E,n+1}$ is the elasticity tensor, and $\delta > 0$ is a Lagrange multiplier chosen so that $\boldsymbol{\epsilon}^{E,n+1}$ is on the yield surface.

Due to our assumption of isotropy, the constraint in Equation (2.24) can be satisfied in terms of the singular values of the elastic deformation gradient. Furthermore, the singular vectors of the trial elastic strain do not change in the return mapping:

$$\mathbf{F}^{E,n+1} = \mathbf{U}^E\boldsymbol{\Sigma}^{E,n+1}(\mathbf{V}^E)^T,\ \tilde{\mathbf{F}}^E = \mathbf{U}^E\tilde{\boldsymbol{\Sigma}}^E(\mathbf{V}^E)^T. \tag{2.25}$$

With this convention, the trial and projected Hencky strains and Kirchhoff stresses satisfy

$$\tilde{\boldsymbol{\epsilon}}^E = \mathbf{U}^E \ln \tilde{\boldsymbol{\Sigma}}^E (\mathbf{U}^E)^T \tag{2.26}$$

$$\boldsymbol{\epsilon}^{E,n+1} = \mathbf{U}^E \ln \boldsymbol{\Sigma}^{E,n+1} (\mathbf{U}^E)^T \tag{2.27}$$

and

$$\tilde{\boldsymbol{\tau}}^E = \mathbf{U}^E(\lambda \mathrm{tr}(\ln(\tilde{\boldsymbol{\Sigma}}^E))\mathbf{I} + 2\mu \ln(\tilde{\boldsymbol{\Sigma}}^E))(\mathbf{U}^E)^T \tag{2.28}$$

$$\boldsymbol{\tau}^{E,n+1} = \mathbf{U}^E(\lambda \mathrm{tr}(\ln(\boldsymbol{\Sigma}^{E,n+1}))\mathbf{I} + 2\mu \ln(\boldsymbol{\Sigma}^{E,n+1}))(\mathbf{U}^E)^T, \tag{2.29}$$

respectively.

The return mapping is completed as an operation on the eigenvalues $\tilde{\boldsymbol{\epsilon}}^E$. For simplicity of notation, we henceforth denote the eigenvalues of $\tilde{\boldsymbol{\epsilon}}^E$ and $\tilde{\boldsymbol{\tau}}^E$ by $\hat{\boldsymbol{\epsilon}}$ and $\hat{\boldsymbol{\tau}} = \lambda(\mathbf{1}\cdot\hat{\boldsymbol{\epsilon}})\mathbf{1} + 2\mu\hat{\boldsymbol{\epsilon}}$ respectively, where $\mathbf{1}$ is the vector of all ones. Furthermore, we refer to the eigenvalues of the projected $\boldsymbol{\epsilon}^{E,n+1}$ and $\boldsymbol{\tau}^{E,n+1}$ as $\mathrm{proj}(\hat{\boldsymbol{\epsilon}})$ and $\mathrm{proj}(\hat{\boldsymbol{\tau}})$ respectively. The process of satisfying Equation (2.24) is, in the case of the Rankine yield condition,

- If $\lambda\mathbf{1}\cdot\hat{\boldsymbol{\epsilon}} + 2\mu\epsilon_1 \le \tau_C$, no projection, $\mathrm{proj}(\hat{\boldsymbol{\epsilon}}) = \hat{\boldsymbol{\epsilon}}$,
- If $(2\mu+\lambda)\epsilon_2 + \lambda(\mathbf{1}\cdot\hat{\boldsymbol{\epsilon}} - \epsilon_1) \le \tau_C < \lambda\mathbf{1}\cdot\hat{\boldsymbol{\epsilon}} + 2\mu\epsilon_1$, $\mathrm{proj}(\hat{\boldsymbol{\epsilon}}) = (\frac{\tau_C - \lambda(\mathbf{1}\cdot\hat{\boldsymbol{\epsilon}}-\epsilon_1)}{2\mu+\lambda}, \epsilon_2, \epsilon_3)$,
- If $(2\mu+3\lambda)\epsilon_3 \le \tau_C < (2\mu+\lambda)\epsilon_2 + \lambda(\mathbf{1}\cdot\hat{\boldsymbol{\epsilon}}-\epsilon_1)$, $\mathrm{proj}(\hat{\boldsymbol{\epsilon}}) = (\frac{\tau_C - \lambda(\mathbf{1}\cdot\hat{\boldsymbol{\epsilon}}-\epsilon_1-\epsilon_2)}{2\mu+2\lambda}, \frac{\tau_C - \lambda(\mathbf{1}\cdot\hat{\boldsymbol{\epsilon}}-\epsilon_1-\epsilon_2)}{2\mu+2\lambda}, \epsilon_3)$,
- If $\tau_C < (2\mu+3\lambda)\epsilon_3$, $\mathrm{proj}(\hat{\boldsymbol{\epsilon}}) = \frac{\tau_C}{2\mu+3\lambda}\mathbf{1}$.

In the case of the von Mises yield condition, the projection is

- If $|\hat{\boldsymbol{\tau}} - \mathbf{1}\cdot\hat{\boldsymbol{\tau}}\mathbf{1}| \le \tau_C$, no projection, $\mathrm{proj}(\hat{\boldsymbol{\epsilon}}) = \hat{\boldsymbol{\epsilon}}$,
- If $|\hat{\boldsymbol{\tau}} - \mathbf{1}\cdot\hat{\boldsymbol{\tau}}\mathbf{1}| > \tau_C$, $\mathbf{p} = (\hat{\boldsymbol{\tau}}\cdot\mathbf{1})\frac{\mathbf{1}}{3}$, $\mathbf{d} = \hat{\boldsymbol{\tau}} - \mathbf{p}$, $\mathrm{proj}(\hat{\boldsymbol{\tau}}) = \mathbf{p} + \tau_C\frac{\mathbf{d}}{|\mathbf{d}|}$, $\mathrm{proj}(\hat{\boldsymbol{\epsilon}}) = \hat{\mathcal{C}}^{-1}\mathrm{proj}(\hat{\boldsymbol{\tau}})$

where

$$\hat{\mathcal{C}} = \begin{pmatrix} 2\mu+\lambda & \lambda & \lambda \\ \lambda & 2\mu+\lambda & \lambda \\ \lambda & \lambda & 2\mu+\lambda \end{pmatrix}. \tag{2.30}$$

After the projection has been done, the singular values of the time t^{n+1} elastic deformation gradient are computed from $\mathbf{\Sigma}^{E,n+1} = \exp(\text{proj}(\hat{\boldsymbol{\epsilon}}))$, which are used to construct the deformation gradient as in Equation (2.25). Lastly, the time t^{n+1} plastic deformation gradient is computed from $\mathbf{F}^{P,n+1} = (\mathbf{F}^{E,n+1})^{-1}\mathbf{F}^{n+1}$.

CHAPTER 3

Thermomechanical Simulation of Baking and Cooking

3.1 Introduction

Whether it is bread rising in the oven, cookies oozing with melting chocolate chips, or a pancake sizzling in a pan, baking and cooking are integral parts of our everyday lives. It is therefore important and yet challenging to model these phenomena accurately when creating compelling virtual scenes for computer graphics applications. Surprisingly, given our everyday familiarity, these processes involve a wide range of complex physical phenomena including heat and mass transfer [NSG14, BT02], viscous and elastic rheology [FF12], dynamics of porous mixtures [DLH10, SKI13] and many more. We develop a model and numerical methods that can capture some of the most characteristic visual aspects of the baking or general cooking process, such as melting, dehydrating, rising, and gelatinization. Furthermore, our approach allows for realistic simulation of user interactions like breaking and folding of the materials at various stages in the cooking process.

We propose a porous thermo-viscoelastoplastic mixture model. Melting effects are captured by a temperature-based change in viscoelastoplastic constitutive laws. During the cooking process water diffuses through the surface of the material according to Fick's law. This allows for effects such as wrinkling and curling of dehydrated fruit, as well as cracking of the top of baked goods [TS98]. Leavening agents are often a predominant source of rising in baked goods, and we focus on the effects of chemical leaveners like baking powder and baking soda. With these agents, carbon dioxide (CO_2) is created through a chemical reaction, which then expands under heat to help the dough to rise [NSG14, ZD06]; this reaction is activated at

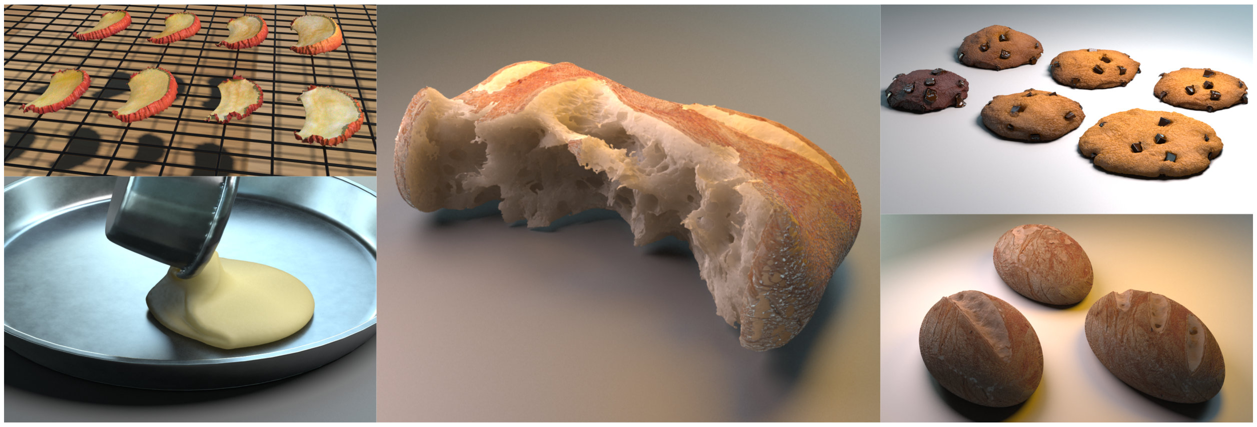


Figure 3.1: **Baking montage**. Representative scenes from some of our simulations. **Top left**: Drying apple slices on a rack. **Bottom left**: Pouring pancake batter into a pan. **Center**: tearing apart a loaf of bread. **Top right**: baking cookies with different amount of leavening agent and under different temperatures. **Bottom right**: baking bread with different/no slits.

the critical temperature, peaks as temperature rises and finally gets deactivated when it goes beyond a threshold. Our model keeps track of the CO_2 creation process to capture the rising effect in baking. At the final stage of the baking process, flour gelatinization takes place and the baked goods become much more elastic and less viscous than the initial dough or batter [VLT09]. This is achieved in our model with temperature-dependent plasticity. We demonstrate these abilities with baking, tearing and dehydrating examples.

While dough and batter are mixtures of constituents that include water, leavening agents, flour, eggs, fat, sugar and others, we model the non-water or non-CO_2 agents as a single phase in a three species mixture model. CO_2 from leavening agents and water make up the other two species in the mixture. We refer to the mixture of non-water and non-CO_2 species as the solid phase and model it as viscoelastoplastic with parameters that vary with temperature. This allows us to address rheological changes induced by cooking that do not arise from the effects of the leavening agents or water-based mass transfer. It is a simplification, but we confidently make it as it reduces the modeling complexity without precluding important features like melting and solidification.

We discretize our model with a novel Material Point Method [SCS94] designed to treat thermomechanical porous mixtures and temperature dependent chemical production of gas from leavening agents. Our approach is designed to track the solid phase (everything but the water and CO_2) since it is most apparent for visual rendering. Various MPM approaches have treated porous mixtures of water, gas and solid species [ASB14, BS15, BFL16, GPH18, TGK17, ZWC09, YAP15, JSV13]. Tampubolon et al. [GPH18, TGK17] use two sets of material points to track distinct water and sand phases. This allows for detailed visual resolution of each phase, but it is costly since each phase must be treated with a separate grid and their interactions are resolved via a stiff interaction term. Our approach is similar to [ASB14, BFL16, YAP15] in that they use one set of material points and track relative motion of the other species. However, these works only considered mixtures of soil, water and air, largely for landslide applications. While there are some similarities in the treatment of porosity, none have considered thermal effects in the solid species as we do. Also, none have modeled gas pressures arising from temperature dependent chemical reactions. We summarize our contributions as

- A thermo-viscoelastoplastic model of dough and batter that approximates mixtures of non-water or CO_2 contents, i.e. ingredients like flour, egg, fat, sugar, etc.
- A three species porous mixture model of the water, CO_2 and remaining dough and batter contents.
- A model for the thermomechanical production of CO_2 from leavening agents and its influence on the viscoelastoplastic rheology of the remaining materials in the solid phase.
- A novel MPM discretization of three species thermomechanical mixtures of solid, gas and water including the effects of chemical production of CO_2 from leavening agents.

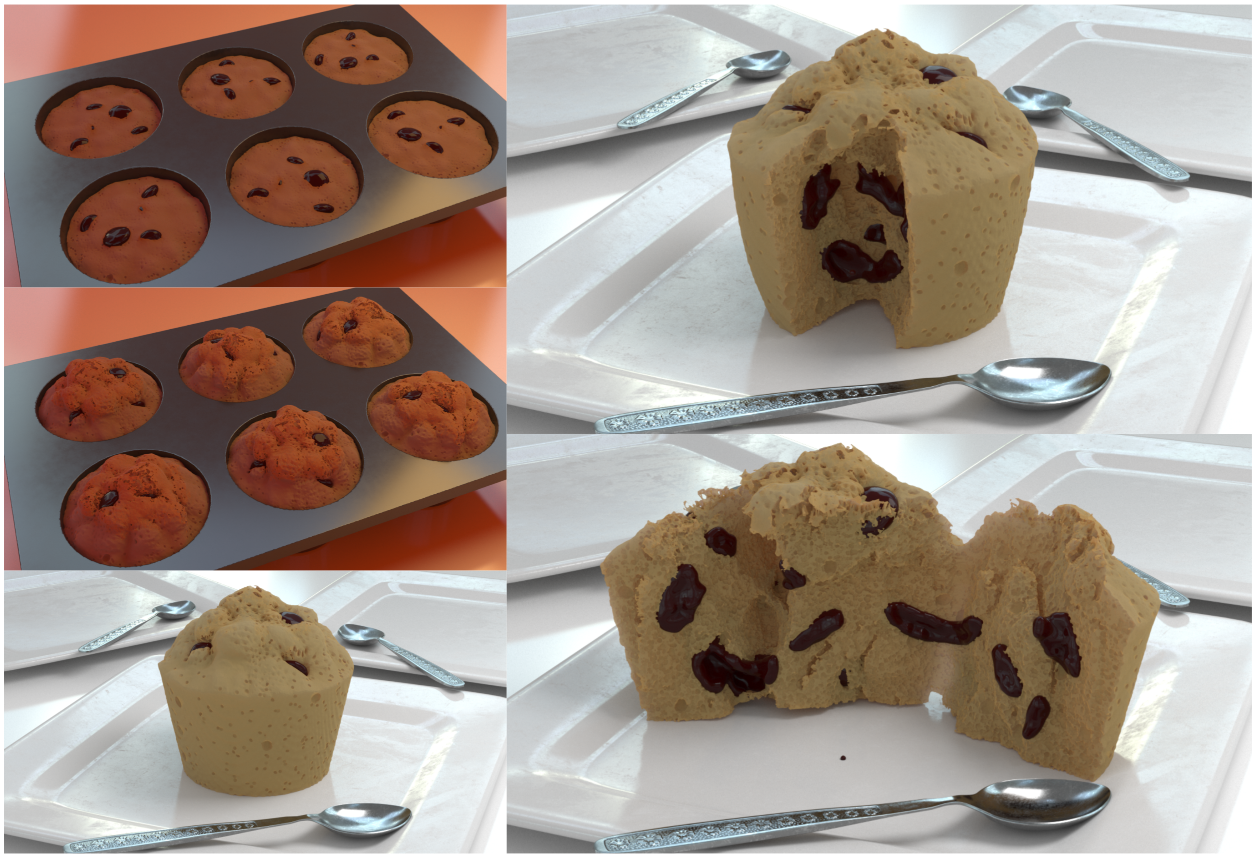


Figure 3.2: **Muffin**. The left column depicts baking of a tray of muffins, resulting in a classic dome shape on top. The right shows a muffin torn open to reveal a fully cooked interior and melted chocolate chips. Surface meshes of the fractured muffin are generated through the mesh post-process of [WDG19] and are used for rendering.

3.2 Previous Work

Many efforts have been made to study one or a few aspects of complicated food processes. Vanin et al. [VLT09] focus on the formation and distinct texture of bread crust due to significantly lower water content than the crumb caused by rapid drying on the surface from the heat of the oven. Guillard et al. [GBG04] model the moisture content evolution in dry biscuit based on Fick's second law [Fic55]. De Cindio and Correra [CC95] view a leavened dough as a visco-elastic homogeneous macrosystem but also a gas-paste microsystem to model the interactions between the two phases during mixing, leavening and baking. Zhang and Datta [ZD06] develop a coupled system of solid, liquid water, water vapor and carbon dioxide phases to model the bread baking process. They model the material as viscoelastic and track the evolution of temperature, moisture, volume and surface coloration during bread baking. Yang et al. [YCL17] develop a unified particle framework for simulating various viscoelastic soft-matter with phase changes using conservative Cahn-Hilliard advection. They produce compelling simulations of phase transitions for cooking eggs and melting butter. Liu et al. [LYC15] use elastoplasticity to simulate porous dehydration of fruits and other foods with the Finite Element Method (FEM).

Graphics applications have used mixture theory and multi-species simulations to model porous water, sand and air mixtures [LWG08, GPH18, TGK17]. Many other graphics applications have made compelling use of multi-species simulations for water, sprays and foams [NO13, TFK03, LTK08, YLH14], liquids with bubbles [MMS09, TSS07, RJL15] as well as for mixing of fluids [BWZ10, KPN10, RLY14, HWZ15, YCR15]. Heat transfer and viscoelastic melting have also been used in many graphics applications to simulate visual effects of phase change and melting [TPF91, MHT05, KAG05, MKN04, THM04, WRK10, LIG06, ZWQ06, WLK03, CMI02, MGG10, GBO04, BWH07, GBB09, WT08, BGA17].

The Material Point Method (MPM) [SCS94] has been used in many computer graphics applications, including snow [SSC13], non-Newtonian fluids and foams [YSB15, RGJ15], heat transfer and phase change [SSJ14, GTJ17], elastic and porous materials [JGT17, GHF18, FBG18], and granular materials [DB16, KGP16, YSC18]. Various MPM approaches have treated porous mixtures of different liquid, gas and solid species. Bandara et al. [ASB14, BS15, BFL16] simulate mixtures of air, water and soil for landslide applications. In [BS15], they use two sets of Lagrangian marker particles for water and soil respectively. Tampubolon et al. [GPH18, TGK17] use a similar approach. In [ASB14, BFL16], they use a single set of material points for the soil and track the motion of the water relative to the soil, but they neglect the water acceleration. Zhang et al. [ZWC09] also neglect the water acceleration terms. Yerro et al. [YAP15] build on the two-phase porous MPM approach of Jassim et al. [JSV13] to simulate three species mixtures of air, water and soil. Bandara et al. [BS15] use Biot's [Bio41] phenomenological model for their governing equations.

3.3 Governing Equations

We model our materials as a mixture of water, gas and solid constituents and use a multi-species continuum model to derive the equations of motion from the governing physics [AC76]. With this assumption we use a different flow map for each constituent to describe the kinematics of the mixture. Formally, we use $\boldsymbol{\phi}^\alpha : B_0 \times [0,T] \to \mathbb{R}^3$ to define the motion of species α where $\alpha = w$ for water, $\alpha = s$ for solid and $\alpha = g$ for gas. Here $B_0 \subset \mathbb{R}^3$ represents the configuration of the material at time $t = 0$. We refer to points $\mathbf{X} \in B_0$ as material points and $\mathbf{x} \in B_t^\alpha = \{\mathbf{x} \in \mathbb{R}^3 | \exists \mathbf{X} \in B_0 \text{ with } \mathbf{x} = \boldsymbol{\phi}^\alpha(\mathbf{X}, t)\}$ as world-space points where B_t^α is the time t configuration of species α. Furthermore, we use

$$\Omega_t^\alpha = \left\{\mathbf{x} \in \mathbb{R}^3 | \exists \mathbf{X} \in \Omega_0 \text{ with } \mathbf{x} = \boldsymbol{\phi}^\alpha(\mathbf{X}, t)\right\} \tag{3.1}$$

for $\Omega_0 \subset B_0$ to denote subsets of the domains. Note that we do not distinguish between different species in B_0 since we assume they are all present in the initial mixture. Furthermore, we assume that the motion of the gas relative to the solid is negligible, and that therefore $\boldsymbol{\phi}^s = \boldsymbol{\phi}^g$ (and $B_t^s = B_t^g$). We also assume that water may diffuse out of the mixture, but

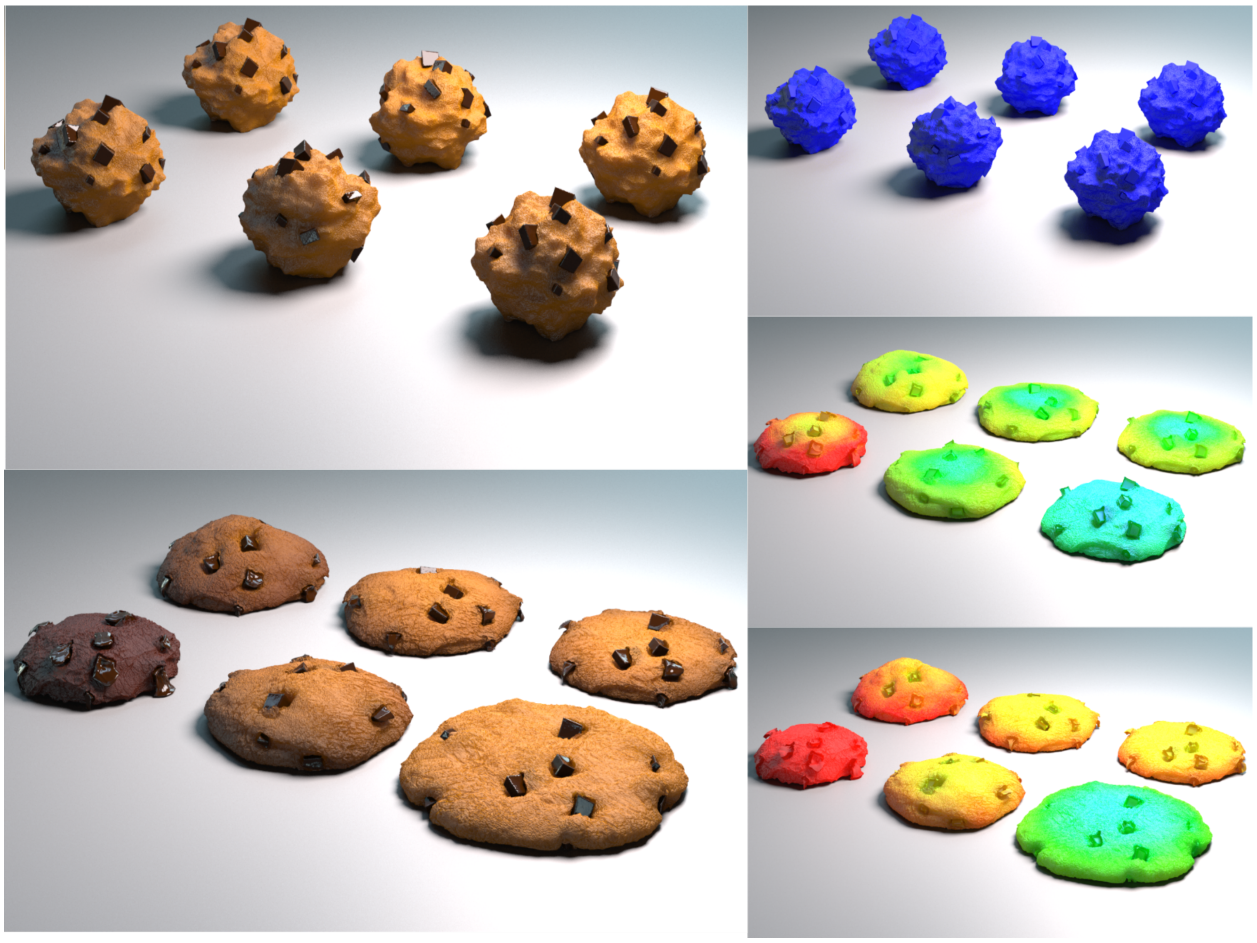


Figure 3.3: **Cookies**. The cookies have a varying amount of leavening agent and are baked under different temperatures. They are initialized as dough balls (**top left**). The **bottom left** shows the end results. The top row from left to right varies with decreasing amounts of leavener, and the bottom row from left to right with decreasing oven temperature. The **right column** depicts the heat transfer progress in the cookies during baking with color varying from blue to green then to red with increasing temperature.

that our mixture will never be completely dry. With this assumption $B_t^s \subset B_t^w$.

The Lagrangian velocity of each constituent is defined as $\mathbf{V}^\alpha(\mathbf{X},t) = \frac{\partial \boldsymbol{\phi}^\alpha}{\partial t}(\mathbf{X},t)$ and the Eulerian (or world-space) velocity is defined as $\mathbf{v}^\alpha(\mathbf{x},t) = \mathbf{V}^\alpha(\boldsymbol{\phi}^{\alpha^{-1}}(\mathbf{x},t),t)$ where $\boldsymbol{\phi}^{\alpha^{-1}}(\mathbf{x},t)$ is the inverse flow map of the species α. We use $\mathbf{F}^\alpha(\mathbf{X},t) = \frac{\partial \boldsymbol{\phi}^\alpha}{\partial \mathbf{X}}(\mathbf{X},t)$ to denote the Jacobian of the constituent mappings (or deformation gradients) and $J^\alpha = \det(\mathbf{F}^\alpha)$ to denote their determinants. Intuitively, the deviation of $\mathbf{F}^\alpha(\mathbf{X},t)$ from orthogonality indicates how non-rigid the motion is local to material point $\mathbf{X}$ and J^α expresses the local volume gain ($J^\alpha > 1$) or loss ($J^\alpha < 1$). Furthermore, we use $\boldsymbol{\phi} : B_0 \times [0,T] \to \mathbb{R}^3$ to denote the flow map of the mixture. It is related to the Eulerian velocity of the mixture $\mathbf{v}$ as $\frac{\partial \boldsymbol{\phi}}{\partial t}(\mathbf{X},t) = \mathbf{v}(\boldsymbol{\phi}(\mathbf{X},t),t)$. The deformation gradient of the mixture and its determinant are denoted as $\mathbf{F}$ and J respectively.

The mass and momentum densities of the species are denoted as $\rho^\alpha : B_t^\alpha \to \mathbb{R}$ and $\rho^\alpha \mathbf{v}^\alpha : B_t^\alpha \to \mathbb{R}$ respectively. The mass and momentum densities of the mixture are defined as the respective sums $\rho = \sum_\alpha \rho^\alpha$ and $\rho \mathbf{v} = \sum_\alpha \rho^\alpha \mathbf{v}^\alpha$. With this convention, the velocity of the mixture is defined via mass average $\mathbf{v} = \frac{\sum_\alpha \rho^\alpha \mathbf{v}^\alpha}{\sum_\alpha \rho^\alpha}$. We assume that the mass density of the gas ρ^g is negligible compared to that of the water and solid and therefore that $\mathbf{v} \approx \frac{\rho^w \mathbf{v}^w + \rho^s \mathbf{v}^s}{\rho^w + \rho^s}$. Furthermore, we assume that the density of water is initially spatially constant and equal to ρ_0^w.

Lastly, we use $\Theta : B_0 \times [0,T] \to \mathbb{R}$ to refer to the material-space temperature of the solid and $\theta : B_t \times [0,T] \to \mathbb{R}$ as the world-space temperature. They are related through $\theta(\boldsymbol{\phi}(\mathbf{X},t),t) = \Theta(\mathbf{X},t)$.

3.3.1 Conservation of Mass and Momentum

Following [AC76], we assume each constituent obeys conservation of mass with respect to its own motion $\frac{D^\alpha \rho^\alpha}{Dt} + \rho^\alpha \nabla \cdot \mathbf{v}^\alpha = 0$ where $\frac{D^\alpha f}{Dt} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial \mathbf{x}} \mathbf{v}^\alpha$ is the material derivative with

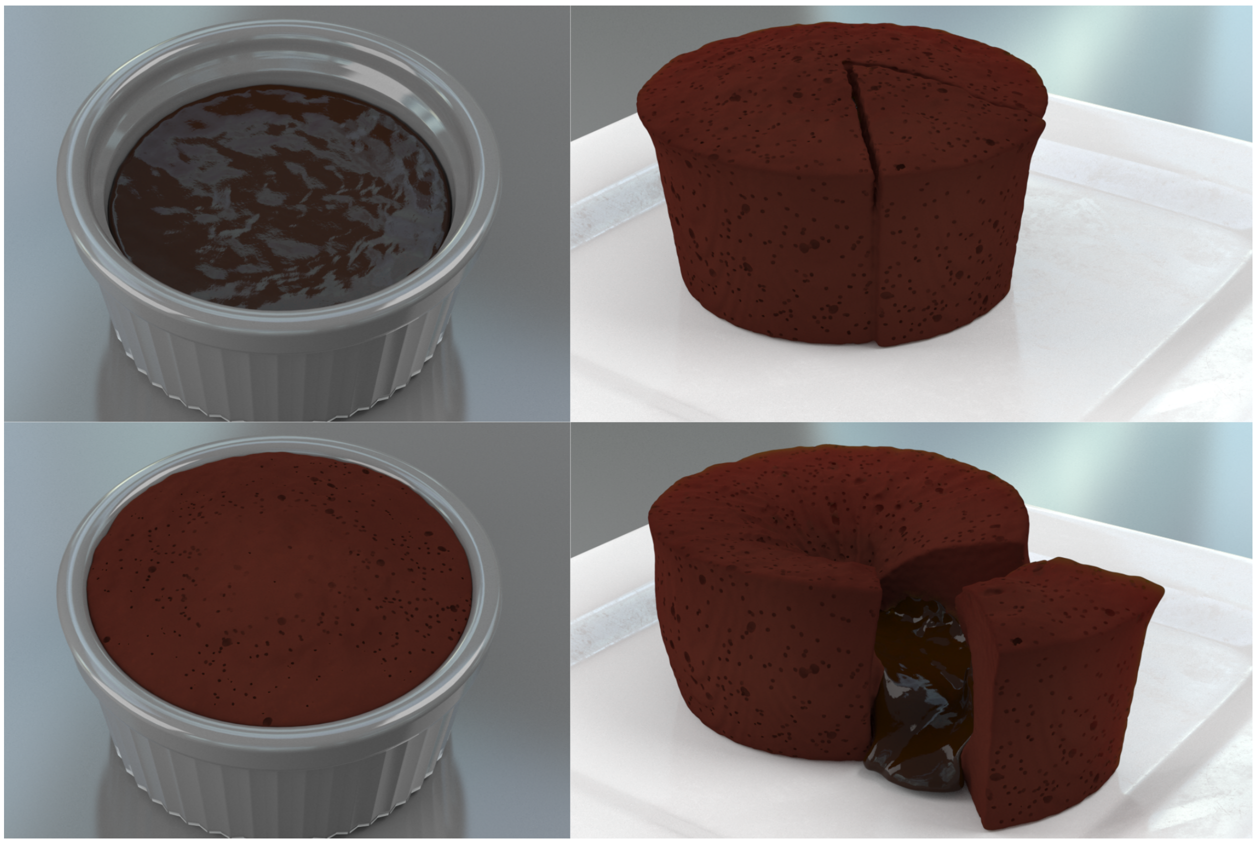


Figure 3.4: **Lava cake**. A lava cake is initialized as a homogeneous batter and baked in a ramekin, then plated and cut open, the molten center flows out.

respect to motion α. This can equivalently by expressed as

$$\frac{d}{dt}\int_{\Omega_t^\alpha} \rho^\alpha d\mathbf{x} = \int_{\Omega_t^\alpha} \frac{d\rho^\alpha}{dt} d\mathbf{x} + \int_{\partial\Omega_t^\alpha} \rho^\alpha \mathbf{v}^\alpha \cdot \mathbf{n} ds(\mathbf{x}) = 0 \tag{3.2}$$

where Ω_t^α is an arbitrary subset of B_t^α.

To simplify our numerical approach it is convenient to adopt the convention of [ZP13] and derive the mass and momentum conservation of the mixture in terms of the motion of the solid constituent. To derive this, we note that the rate of change of the mass of water in Ω_t^s due to motion of the solid is

$$\frac{d}{dt}\int_{\Omega_t^s} \rho^w d\mathbf{x} = \int_{\Omega_t^s} \frac{d\rho^w}{dt} d\mathbf{x} + \int_{\partial\Omega_t^s} \rho^w \mathbf{v}^s \cdot \mathbf{n} ds(\mathbf{x}). \tag{3.3}$$

Here the time derivative takes into account the change in the set Ω_t^s and its effect as the domain of integration. Consider the case when $\Omega_t^w = \Omega_t^s$ at a given time t. In this case the sets are equal, but may not be so for earlier times $\hat{t} < t$, e.g. $\Omega_{\hat{t}}^w \neq \Omega_{\hat{t}}^s$, when we consider how they evolve under the respective constituent motions via Equation (3.1). We can subtract Equation (3.2) (in the case of $\alpha = w$) from Equation (3.3) to yield an expression of conservation of water mass with respect to motion of the solid

$$\frac{d}{dt}\int_{\Omega_t^s} \rho^w d\mathbf{x} + \int_{\partial\Omega_t^s} \rho^w (\mathbf{v}^w - \mathbf{v}^s) \cdot \mathbf{n} ds(\mathbf{x}) = 0.$$

We henceforth use $\mathbf{q}^w = \rho^w(\mathbf{v}^w - \mathbf{v}^s)$ to denote the density weighted velocity of the water relative to the solid. Since the set Ω_t^s is arbitrary, conservation of mass of the water with respect to the motion of the solid can be expressed as

$$\frac{D^s \rho^w}{Dt} + \rho^w \nabla \cdot \mathbf{v}^s + \nabla \cdot \mathbf{q}^w = 0, \ \mathbf{x} \in B_t^s. \tag{3.4}$$

Momentum balance for each constituent can similarly be expressed as

$$\frac{d}{dt}\int_{\Omega_t^\alpha} \rho^\alpha \mathbf{v}^\alpha d\mathbf{x} = \int_{\Omega_t^\alpha} (\mathbf{f}^\alpha + \mathbf{p}^\alpha) d\mathbf{x} + \int_{\partial\Omega_t^\alpha} \mathbf{t}^\alpha ds(\mathbf{x})$$

where $\mathbf{f}^\alpha$ and $\mathbf{t}^\alpha$ are the external body force and traction on phase α, and the $\mathbf{p}^\alpha$ term stands for the transfer of momentum between the constituents. We refer the reader to [AC76] for a more detailed derivation and motivation of these balances and the momentum exchanges between them. We note that

$$\frac{d}{dt}\int_{\Omega_t^\alpha} \rho^\alpha \mathbf{v}^\alpha d\mathbf{x} = \int_{\Omega_t^\alpha} \frac{d}{dt}(\rho^\alpha \mathbf{v}^\alpha) d\mathbf{x} + \int_{\partial\Omega_t^\alpha} \rho^\alpha \mathbf{v}^\alpha \otimes \mathbf{v}^\alpha \cdot \mathbf{n} ds(\mathbf{x}). \tag{3.5}$$

As in the case of water mass, the rate of change of water momentum with respect to the motion of the solid can be expressed as

$$\frac{d}{dt}\int_{\Omega_t^s} \rho^w \mathbf{v}^w d\mathbf{x} = \int_{\Omega_t^s} \frac{d}{dt}(\rho^w \mathbf{v}^w) d\mathbf{x} + \int_{\partial\Omega_t^s} \rho^w \mathbf{v}^w \otimes \mathbf{v}^s \cdot \mathbf{n} ds(\mathbf{x}). \tag{3.6}$$

Again as with water mass, by considering coincident sets at time t, $\Omega_t^s = \Omega_t^w$ and subtracting Equation (3.5) (in the case of $\alpha = w$) from Equation (3.6), we can conclude

$$\frac{d}{dt}\int_{\Omega_t^s} \rho^w \mathbf{v}^w d\mathbf{x} = \frac{d}{dt}\int_{\Omega_t^w} \rho^w \mathbf{v}^w d\mathbf{x} - \int_{\partial\Omega_t^s} \mathbf{v}^w \otimes \mathbf{q}^w \cdot \mathbf{n} ds(\mathbf{x}).$$

We can use this equality to express conservation of momentum of the mixture relative to the motion of the solid constituent as

$$\begin{aligned}
&\frac{d}{dt}\int_{\Omega_t^s} \rho \mathbf{v} d\mathbf{x} \\
=&\frac{d}{dt}\int_{\Omega_t^s} (\rho^s \mathbf{v}^s + \rho^w \mathbf{v}^w) d\mathbf{x} \\
=&\frac{d}{dt}\int_{\Omega_t^s} \rho^s \mathbf{v}^s d\mathbf{x} + \frac{d}{dt}\int_{\Omega_t^w} \rho^w \mathbf{v}^w d\mathbf{x} - \int_{\partial\Omega_t^s} \mathbf{v}^w \otimes \mathbf{q}^w \cdot \mathbf{n} ds(\mathbf{x}) \\
=&\int_{\Omega_t^s} (\mathbf{f}^s + \mathbf{p}^s) d\mathbf{x} + \int_{\partial\Omega_t^s} \mathbf{t}^s ds(\mathbf{x}) + \int_{\Omega_t^w} (\mathbf{f}^w + \mathbf{p}^w) d\mathbf{x} \\
&+ \int_{\partial\Omega_t^s} \mathbf{t}^w ds(\mathbf{x}) - \int_{\partial\Omega_t^s} \mathbf{v}^w \otimes \mathbf{q}^w \cdot \mathbf{n} ds(\mathbf{x}) \\
=&\int_{\Omega_t^s} \mathbf{f} d\mathbf{x} + \int_{\partial\Omega_t^s} \mathbf{t} ds(\mathbf{x}) - \int_{\partial\Omega_t^s} \mathbf{v}^w \otimes \mathbf{q}^w \cdot \mathbf{n} ds(\mathbf{x})
\end{aligned}$$

where $\mathbf{f} = \mathbf{f}^s + \mathbf{f}^w$ is the total body force on the mixture, and $\mathbf{t} = \mathbf{t}^s + \mathbf{t}^w$ is the total traction and the sum of the momentum exchange terms $\mathbf{p}^w + \mathbf{p}^s = \mathbf{0}$ is assumed to be zero [AC76]. Therefore, since the set $\Omega_t^s \subset B_t^s$ is arbitrary, we can conclude that conservation of mass of the mixture can be expressed with respect to motion of the solid constituent as

$$\frac{D^s \rho \mathbf{v}}{Dt} + \rho \mathbf{v} \nabla \cdot \mathbf{v}^s = \mathbf{f} + \nabla \cdot \boldsymbol{\sigma} - \nabla \cdot (\mathbf{v}^w \otimes \mathbf{q}^w), \ \mathbf{x} \in B_t^s. \tag{3.7}$$

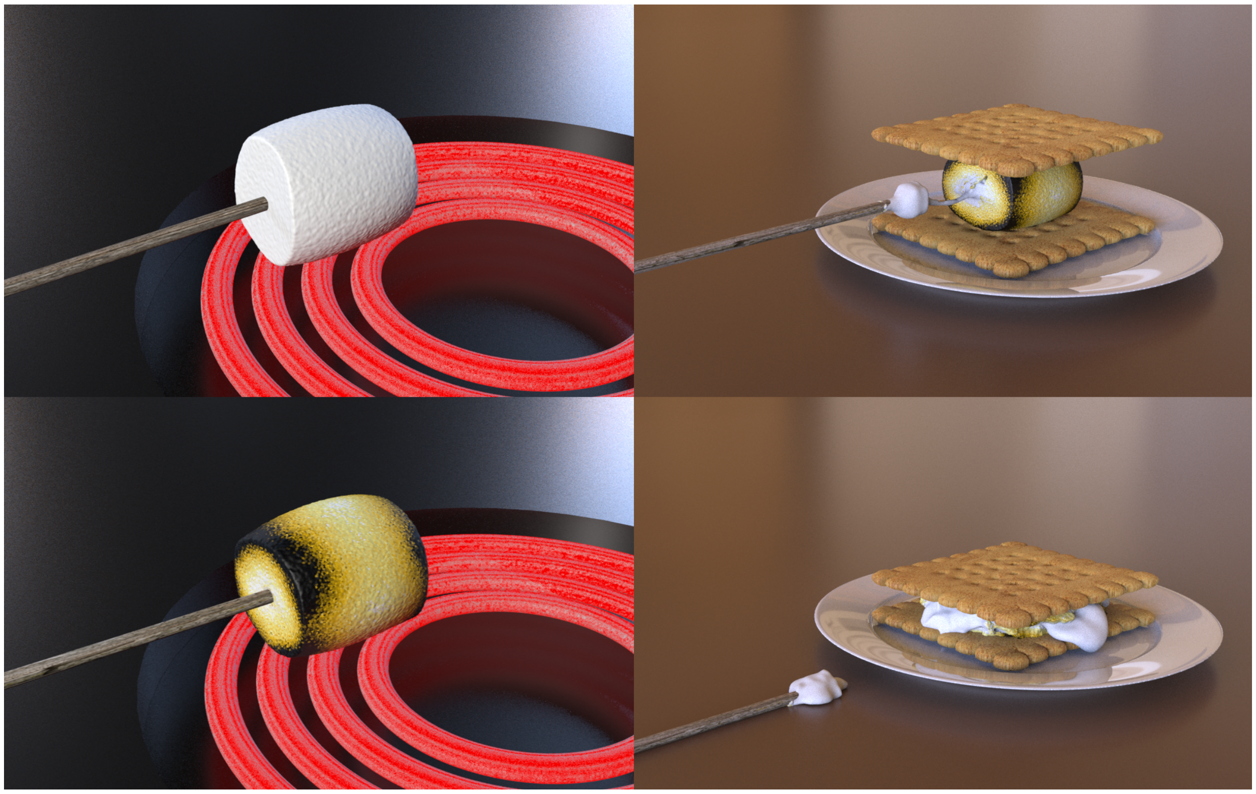


Figure 3.5: **S'more**. The **left column** depicts a marshmallow roasting on the electric stove. Coloring is based on temperature. We press the marshmallow between two crackers (**right column**) and the melted interior flows out.

Here $\boldsymbol{\sigma}$ is the Cauchy stress in the mixture and it is related to the total traction of the mixture as $\mathbf{t} = \boldsymbol{\sigma}\mathbf{n}$.

3.3.2 Fick's Law

We assume that the motion of the liquid phase is restricted to diffusion in the mixture; Fick's law [Fic55] states that

$$\mathbf{q}^w = -K^w \theta \rho^w \nabla s$$

where $K^w > 0$ is a material diffusivity constant, and $s = \frac{V^w}{V}$ stands for the saturation level of the dough, which is in the form of a volume fraction where V is a local volume of the mixture and V^w is the local volume of water in the mixture. The density of water can be expressed in terms of this saturation as

$$\rho^w = \frac{m^w}{V} = \frac{m^w}{V^w}\frac{V^w}{V} = \rho_0^w s. \tag{3.8}$$

Furthermore, we use $\mathbf{q}^w \cdot \mathbf{n} = \beta^w \theta s$ for the boundary conditions in Equation (3.4) where β^w is a material constant that represents the rate of water mass loss due to exterior conditions.

3.4 Heat Transfer

Heat flow through the material obeys the first and second law of thermodynamics, in Section 1.1.2.3 we derived the general heat balance equation. Combined with our choice of potential we summarize our heat energy balance equation as

$$\rho\alpha\frac{D\theta}{Dt} = \nabla \cdot (\kappa\nabla\theta) - \theta c\lambda\nabla\cdot\mathbf{v},\ \mathbf{x} \in B_t. \tag{3.9}$$

Here, α is the specific heat at constant volume which controls the rate at which temperature will change for a given amount of heating and κ is the thermal conductivity which controls the speed with which heat will diffuse through the body. $c\lambda$ controls rate of temperature decrease resulting from thermal expansion and λ is a Lamé coefficient. Intuitively, since temperature measures the kinetic energy density at small scales, increases in material volume will tend to cool the material. The heat flow boundary condition is

$$\mathbf{q}\cdot\mathbf{n} = -\kappa\nabla\theta\cdot\mathbf{n} = \beta(\theta - \theta_{out}),\ \mathbf{x} \in \partial B_t$$

where $\mathbf{q}$ is the heat flux, $\mathbf{n}$ is the outward normal, θ_{out} is the ambient temperature outside of the material and β controls the rate of temperature change that results from heat transfer with the ambient space.

3.5 Constitutive Model

Again following [AC76], the stress in the mixture is a sum of the stresses in the solid, gas and water constituents

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^s + \boldsymbol{\sigma}^g + \boldsymbol{\sigma}^w.$$

3.5.1 Solid Stress

We assume that the solid phase is itself a complex mixture of constituents which may include flour, fat, leavening agent (baking powder, yeast etc.), salt, sugar, gelatin or egg. These assorted ingredients create a mixture material with a wide range of complex, non-Newtonian rheological behaviors whose properties vary from nearly liquid to nearly solid. Furthermore, these behaviors vary with temperature. Ideally, we would include each phase in our mixture model, however to reduce complexity we instead adopt a temperature dependent viscoelasto-plasticity model for the solid phase. This allows us to efficiently reproduce a wide range of complex rheological behaviors without the need to track each of the many species.

3.5.2 Elastic Stress

We use a multiplicative decomposition of the deformation in the solid phase $\mathbf{F}^s = \mathbf{F}^{s,E}\mathbf{F}^{s,P}$ where $\mathbf{F}^{s,E}$ is the elastic component of the motion that will be penalized by the potential energy density and $\mathbf{F}^{s,P}$ is the permanent deformation associated with the plasticity. The stress in the solid phase arises from the potential energy density and thermal expansion as

$$\boldsymbol{\sigma}^s = \frac{\boldsymbol{\tau}^{s,E}}{J^s} - c\lambda J^s \theta. \tag{3.10}$$

where $\boldsymbol{\tau}^{s,E}$ is the elastic Kirchhoff stress defined in terms of the elastic potential as $\boldsymbol{\tau}^{s,E} = \frac{\partial \psi}{\partial \mathbf{F}^{s,E}} \mathbf{F}^{s,E^{-T}}$. The last term is due to thermal expansion and we refer the reader to [GS08] for its derivation. We note that the elastic potential will ultimately have temperature dependence from the thermal effects of plasticity (see Section 3.5.3). Our choice of potential is

$$\psi(\mathbf{F}^{s,E}) = \frac{1}{2}\lambda(\mathrm{tr}(\boldsymbol{\epsilon}))^2 + \mu\boldsymbol{\epsilon} : \boldsymbol{\epsilon} \tag{3.11}$$

where $\boldsymbol{\epsilon}$ is the elastic Hencky strain in the solid $\boldsymbol{\epsilon} = \frac{1}{2}\log(\mathbf{F}^{s,E}\mathbf{F}^{s,E^T})$. Here μ and λ are the Lamé coefficients and $c\lambda$ controls the amount of pressure due to thermal expansion where c is the same parameter that appears in Equation (3.9). Note that the $c\lambda$ term gives rise to a temperature varying positive (expansional) pressure. We also note that our primary motivation for adopting this potential is to simplify discrete plastic integration as in [KGP16, GGT18]. This simplicity is a consequence of its property $\boldsymbol{\tau}^{s,E} = \mathbf{C}\boldsymbol{\epsilon}$ where $\mathbf{C} = 2\mu\mathcal{I} + \lambda\mathbf{I}\otimes\mathbf{I}$ is the isotropic fourth order elastic stiffness tensor and $\mathcal{I}$ is the fourth order identity tensor.

3.5.3 Viscoplasticity

While the potential energy varies with the elastic portion of the decomposition, the evolution of the plastic portion is defined in terms of a yield condition that identifies states of stress consistent with observed material behavior. We represent the yield condition in terms of the Kirchhoff stress $\boldsymbol{\tau} = J\boldsymbol{\sigma}$ and a signed distance function $f(\boldsymbol{\tau})$ where $f(\boldsymbol{\tau}) \leq 0$ indicates a state of physically meaningful stress. We express this evolution of the plastic flow and its relation to the yield condition in terms of the left Cauchy-Green strain $\mathbf{b}^E = \mathbf{F}^{s,E}\mathbf{F}^{s,E^T}$

$$\frac{D^s\mathbf{b}^E}{Dt} = \frac{\partial\mathbf{v}^s}{\partial\mathbf{x}}\mathbf{b}^E + \mathbf{b}^E\frac{\partial\mathbf{v}^s}{\partial\mathbf{x}}^T + \mathcal{L}_{\mathbf{v}^s}\mathbf{b}^E. \tag{3.12}$$

Here $\mathcal{L}^s_{\mathbf{v}}\mathbf{b}^E$ is the Lie derivative with respect to the motion of the solid and is defined in terms of the yield condition. We use an associative flow rule $\mathcal{L}_{\mathbf{v}}\mathbf{b}^E = -2\gamma\frac{\partial f}{\partial\boldsymbol{\tau}}\mathbf{b}^E$ where γ is the flow rate, and $\frac{\partial f}{\partial\boldsymbol{\tau}}\mathbf{b}^E$ specifies the direction of maximum energy dissipation. For rate independent plasticity γ is chosen so that the stress satisfies the yield condition $f(\boldsymbol{\tau}) \leq 0$. In the case of viscoplasticity we do not restrict the stress to always satisfy the yield condition; instead we choose the rate as

$$\gamma = \frac{1}{\eta}\frac{\partial g}{\partial f}(f(\boldsymbol{\tau}))$$

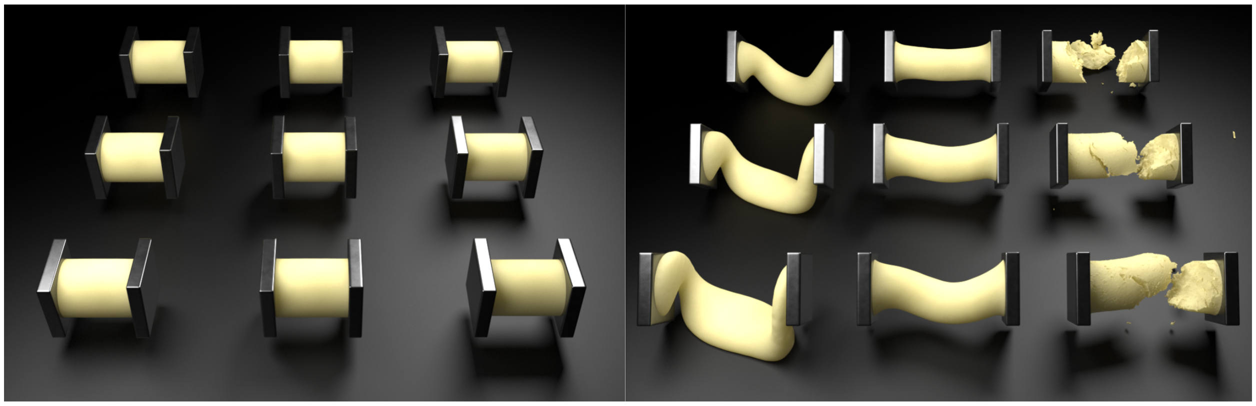


Figure 3.6: **Dough pull and twist**. We apply the same pull and twist motion to dough cylinders of varying yield stress and viscosity parameters. The viscosity increases from bottom to top, and yield stress increases from left to right.

where $\eta = \eta(\theta)$ is a viscosity penalty parameter, $g(f)$ is a monotonic function for positive f and zero otherwise. It is chosen to penalize states of stress outside the yield surface without making adherence to the condition of a hard constraint. We use $g(f) = \frac{1}{2}f^2$ whenever f is positive, and zero otherwise. Note that there is no plastic flow when the stress is inside the yield surface since in this case $f(\boldsymbol{\tau}) \leq 0$ and therefore $\gamma = 0$ since the argument to $\frac{\partial g}{\partial f}$ is non-positive. Lastly, we also note that as $\eta \to 0$, γ is chosen as in the rate independent case. We demonstrate the effect of different plasticity parameters with a dough pull and twist example, see Figure 3.6.

3.5.3.1 Yield Condition

We use a modified temperature and porosity dependent Cam clay yield condition [RB68]. This model is typically used for porous viscoelastoplastic materials and gives rise to a wide range of behaviors similar to those exhibited by the solid phase mixture for baking materials. This yield condition is ellipsoidal in principle stress space (Figure 3.7) and is given by

$$f(\boldsymbol{\tau}) = \frac{9}{4\ln^2\left(\frac{1}{n^p}\right)\left(\frac{7}{2}n^p + 1\right)}p^2 + q^2 - \tau_y(\theta)^2 \leq 0 \tag{3.13}$$

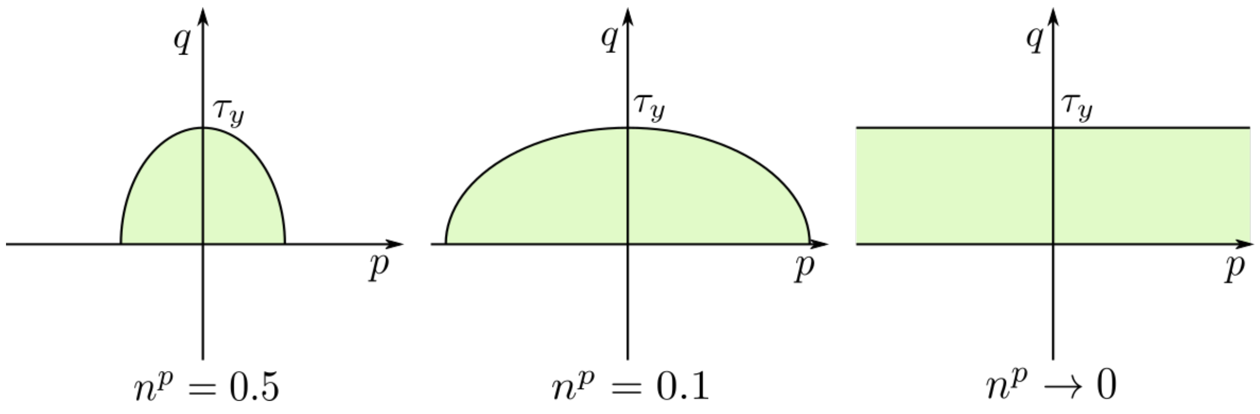
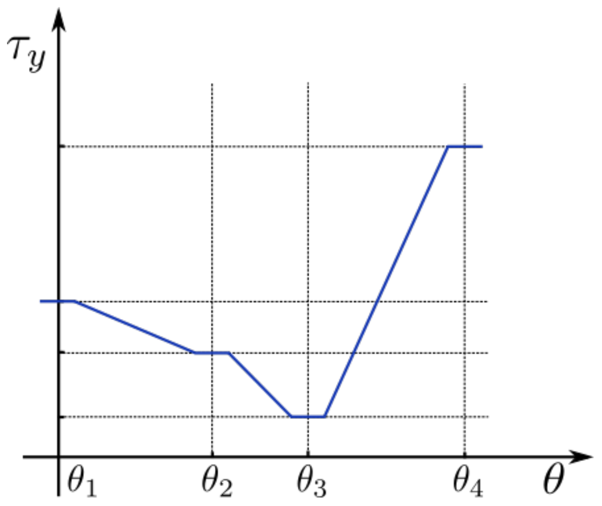


Figure 3.7: **Yield surface**. We visualize the yield surface with different choices of porosity. The yield surface is an ellipsoidal shape in the upper half-plane. In the limit $n^g \to 0$, the yield conditions becomes equivalent to a von Mises type criterion.

where $p = -\frac{1}{3}\mathrm{tr}(\boldsymbol{\tau})$ is the mean normal stress, $q = \sqrt{\frac{3}{2}}\|\boldsymbol{\tau} + p\mathbf{I}\|_F$ is the effective stress (proportional to the magnitude of deviatoric stress) and $\tau_y(\theta)$ is the yield stress. $n^p \in (0,1)$ is the gas porosity of the solid-gas mixture (see Section 3.5.4). We adopt the model of [HA92] where the coefficient of p is a monotonic increasing function of n^p and vise versa. Note that in the limit $n^p \to 0$, the yield function becomes $f(\boldsymbol{\tau}) = q^2 - \tau_y^2 \leq 0$, which is a von Mises type yield criterion equivalent to the viscoplastic formulation of [YSB15]. Intuitively, a more porous solid mixture ($n^p > 0$) will have limits on the degree to which it can achieve cohesion. This cohesion limit is expressed in terms of the tip of the ellipsoid on the negative portion of the p axis.

3.5.3.2 Temperature Dependence

We model the viscosity parameter η and yield stress τ_y as piece-wise linear functions of temperature in order to track the melting and gelatinization of the dough mixture that are often observed during baking. In the initial stage of heating, the fat in the dough, if present, would soften and melt, causing the mixture to appear less viscous and more inelastic. Flour gelatinization takes place later on which increases the elastic strength of the mixture, and



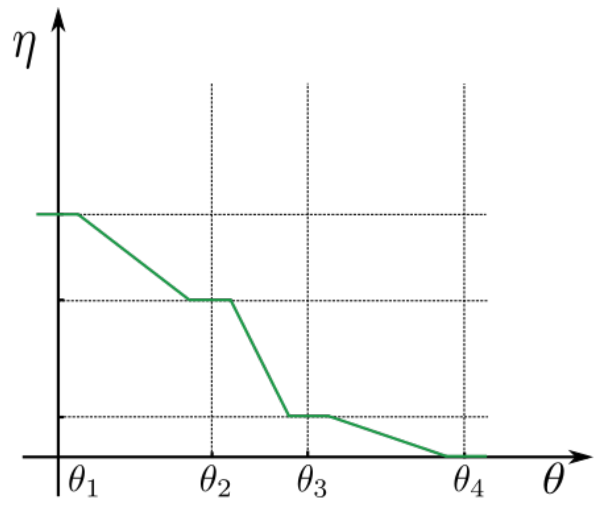


Figure 3.8: **Temperature-dependent plasticity**. We plot the change in yield stress (**left**) and viscosity (**right**) with temperature. The marked temperatures $\theta_1, \theta_2, \theta_3, \theta_4$ stand for typical temperature ranges for fridge, room environment, fat melting point, and starch gelatinization respectively.

this process cannot be reversed during baking. Based on this intuition we set η to be a non-negative monotonic non-increasing function of θ, and τ_y decreasing initially then increasing with respect to temperature, as shown in Figure 3.8. Furthermore, once the material reaches the gelatinization temperature, the parameters are fixed. We follow this progression of plasticity parameters with respect to temperature during baking. The actual values vary among the different examples for optimal visual effects.

3.5.4 Gas Stress

We model the effect of the leavening agents in the mixture with a chemical reaction that creates carbon dioxide (CO_2). The CO_2 then expands under heating to drive the rising of the mixture during the baking process. We model the chemical reaction with the differential equation [ZD06, NSG14]

$$\frac{dn^g}{dt} = \alpha^g R_0 \exp\left(-\left(\frac{\theta - \theta_r}{\Delta\theta}\right)^2\right).$$

Here n^g stands for the molar count of CO_2 per unit volume, α^g is a material constant, R_0 is initial mass density of the mixture, θ_r is a reference temperature at which the chemical reaction reaches its peak rate (Figure 3.12), and $\Delta\theta$ is another constant scaling parameter. Note that the chemical reaction rate is maximal when temperature is around θ_r.

We define the gas porosity of the solid material as $n^p = \frac{V^g}{V^s}$, where V^g stands for a local representative gas volume, and V^s is the total local representative volume of the solid/gas mixture. We assume that porosity changes as the solid mixture flows plastically as

$$n^p = J^{s,P} - 1 + n_0^p$$

where n_0^p is the initial gas porosity. Intuitively, as the material is kneaded it will gain volume plastically, allowing more room for gas. With these conventions, $\frac{n^g}{J^s n^p}$ measures the amount of gas in the pore volume. Therefore, using the ideal gas law to model the pressure of CO_2, the stress in the gas constituent is

$$\boldsymbol{\sigma}^g = -\frac{n^g R\theta}{J^s n^p}\mathbf{I} \tag{3.14}$$

where R is the ideal gas constant.

3.5.5 Water Stress

We adopt the saturation-based pressure model of [ZP13] for the stress in the water phase. As the water leaves a region of the material, the mixture experiences a negative pressure and will tend to contract inwards as the elastic stress is allowed to compress the mixture in the absence of liquid. When water enters a region, the material will experience a positive pressure as the liquid pushes on the mixture. This is modeled with the linear relation

$$\boldsymbol{\sigma}^w = -\hat{c}(s - s_0)\mathbf{I} \tag{3.15}$$

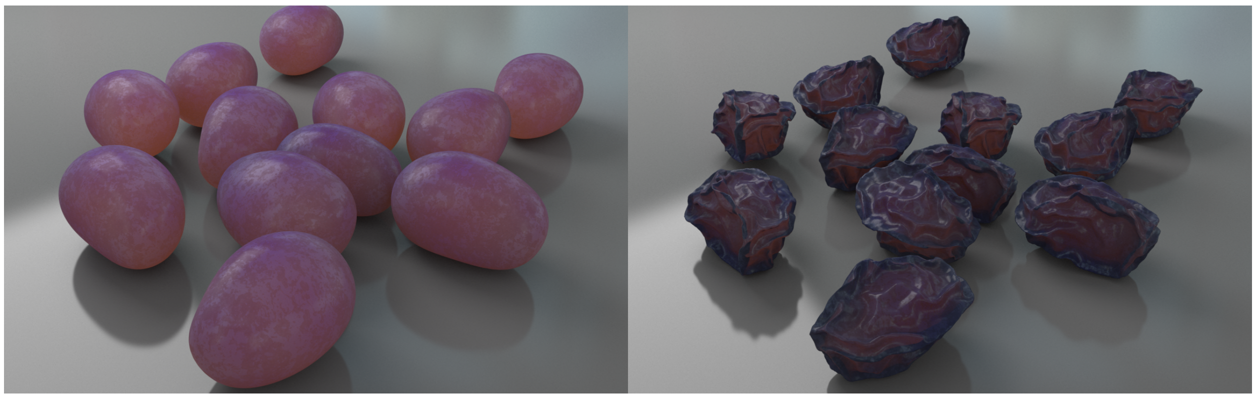


Figure 3.9: **Dehydrating grapes**. Grapes are dried at constant temperature. The smooth exterior (**left**) wrinkles up due to water loss of the interior flesh(**right**).

where s_0 is the initial saturation.

3.6 Discretization

Our discretization is designed to track the solid constituents. We store $\mathbf{x}_p^n = \boldsymbol{\phi}^s(\mathbf{X}_p, t^n)$ to denote the time t^n location of discrete material particle $\mathbf{X}_p$ under the solid motion and an initial representative volume for each particle $V_p^{s,0}$ that partitions the initial domain of the solid. Additionally, we store the translational $\mathbf{v}_p^n = \frac{\partial \boldsymbol{\phi}^s}{\partial t}(\mathbf{X}_p, t^n)$, and affine $\mathbf{A}_p^n \approx \frac{\partial \mathbf{v}^s}{\partial \mathbf{x}}(\mathbf{x}_p^n, t^n)$ velocities and elastic/plastic decomposition of the deformation gradient $\mathbf{F}_p^{s,En}\mathbf{F}_p^{s,Pn} = \mathbf{F}_p^{s,n} = \frac{\partial \boldsymbol{\phi}^s}{\partial \mathbf{X}}(\mathbf{X}_p, t^n)$ of the solid phase. We also store the water $m_p^{w,n}$ and solid m_p^s masses. We note that the solid mass does not change with time since we track the solid motion in a Lagrangian manner. We also note that the water mass satisfies $m_p^{w,n} = \rho^w(\mathbf{x}_p^n, t^n)V_p^{s,n}$ where $V_p^{s,n} = J_p^{s,n}V_p^{s,0}$ is a representative volume around $\mathbf{x}_p^n$ in the time t^n configuration of the solid and $J_p^{s,n} = \det(\mathbf{F}_p^{s,n})$. We store the water saturation s_p^n and the density weighted velocity of the water relative to the solid $\mathbf{q}_p^{w,n}$. We also store the temperature θ_p^n and molar count per volume n_p^g.

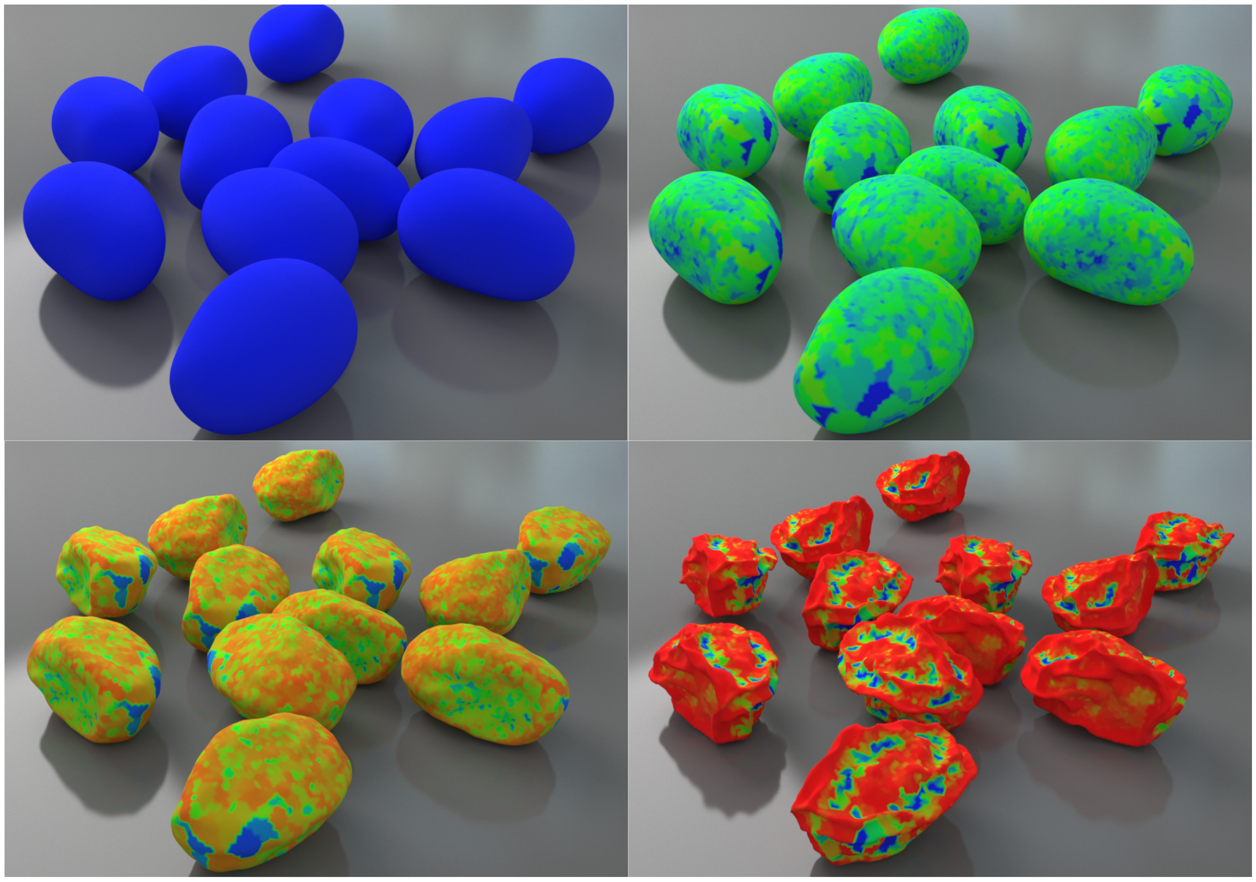


Figure 3.10: **Grape saturation view**. We visualize the saturation evolution of the grapes. The color changes from blue to green then to red with decreasing saturation.

3.6.1 Time Step

We update the discrete state from time t^n to t^{n+1} by discretizing the governing equations with MPM. This requires transferring various data from particles to grid using grid based interpolating functions. We use quadratic B-splines as in [JGT17]. We let $N_{\mathbf{i}}(\mathbf{x})$ denote the grid based interpolating function associated with grid node $\mathbf{x}_{\mathbf{i}}$ and $w_{\mathbf{i}p} = N_{\mathbf{i}}(\mathbf{x}_p^n)$, $\nabla w_{\mathbf{i}p} = \nabla N_{\mathbf{i}}(\mathbf{x}_p^n)$ for short. We transfer the mass, velocity, temperature and particle representative volume to the grid. The momentum is transferred using APIC [JSS15]

$$m_{\mathbf{i}}^{s,n} = \sum_p m_p^s w_{\mathbf{i}p},\ m_{\mathbf{i}}^{w,n} = \sum_p m_p^{w,n} w_{\mathbf{i}p} \tag{3.16}$$

$$\mathbf{v}_{\mathbf{i}}^{s,n} = \frac{1}{m_{\mathbf{i}}^{s,n}} \sum_p m_p^s w_{\mathbf{i}p} \left(\mathbf{v}_p^{s,n} + \mathbf{A}_p^n(\mathbf{x}_{\mathbf{i}} - \mathbf{x}_p^n)\right) \tag{3.17}$$

$$m_{\mathbf{i}}^{s,n}\theta_{\mathbf{i}}^n = \sum_p m_p^{s,n} w_{\mathbf{i}p}\theta_p^n \tag{3.18}$$

$$V_{\mathbf{i}}^{s,n} = \sum_p J_p^{s,n} V_p^{s,0} w_{\mathbf{i}p} \tag{3.19}$$

Here $m_{\mathbf{i}}^{\alpha,n}$ are grid masses, $\mathbf{v}_{\mathbf{i}}^{s,n}$ is grid solid velocity and $\theta_{\mathbf{i}}^n$ is grid temperature. $V_{\mathbf{i}}^{s,n}$ is the representative volume of grid node i which we use to update grid saturation (see Section 3.6.4). We update the grid water mass and temperature from Equations (3.4) and (3.9) respectively to obtain $\tilde{m}_{\mathbf{i}}^{w,n+1}$ and $\theta_{\mathbf{i}}^{n+1}$. We provide the details for these updates in Sections 3.6.4 and 3.6.5 respectively. Once these are updated, we update the particle mass, saturation, temperature and density weighted velocity of the water relative to the solid from

$$m_p^{w,n+1} = m_p^{w,n}\left(1 + \sum_{\mathbf{i}} \frac{\Delta m_{\mathbf{i}}^{w,n}}{m_{\mathbf{i}}^{w,n}} w_{\mathbf{i}p}\right) \tag{3.20}$$

$$s_p^{n+1} = \frac{m_p^{w,n+1}}{V_p^{s,n}\rho_0^w} \tag{3.21}$$

$$\theta_p^{n+1} = \theta_p^{n+1} + \sum_{\mathbf{i}} \left(\theta_{\mathbf{i}}^{n+1} - \theta_{\mathbf{i}}^n\right) w_{\mathbf{i}p} \tag{3.22}$$

$$\mathbf{q}_p^{w,n+1} = -k\theta_p^{n+1} \sum_{\mathbf{i}} s_{\mathbf{i}}^{n+1} \nabla w_{\mathbf{i}p} \rho_0^w s_p^{n+1} \tag{3.23}$$

Here $\Delta m_{\mathbf{i}}^{w,n}$ is from Equation (3.28) and represents the change in the grid water mass and $s_{\mathbf{i}}^{n+1}$ in Equation (3.23) is the time t^{n+1} saturation of grid node $\mathbf{i}$ and is used to compute saturation gradient needed for the update of the density weighted velocity of the water relative to the solid. The weighting of the transfer in Equation (3.21) is chosen so that the total change in particle water mass is equal to the total change in grid water mass. Also, the update in Equation (3.22) follows from Equation (3.8) since $\rho_p^{w,n+1} = \frac{m_p^{w,n+1}}{V_p^{s,n}}$, $\rho_p^{w,n+1} = s_p^{n+1}\rho_0^w$ where ρ_0^w is the initial, spatially constant water mass density.

The last step in our update is to compute the updated grid solid velocity $\tilde{\mathbf{v}}_{\mathbf{i}}^{s,n+1}$ from the discrete conservation of momentum which is outlined in Section 3.6.2. We then compute the updated particle constant $\mathbf{v}_p^{n+1}$ and linear $\mathbf{A}_p^{n+1}$ from $\tilde{\mathbf{v}}_{\mathbf{i}}^{s,n+1}$ using APIC [JSS15] and the particle positions as $\mathbf{x}_p^{n+1} = \mathbf{x}_p^{n+1} + \Delta t \mathbf{v}_p^{n+1}$. Lastly, a trial state of elastic stress is computed assuming no plastic flow over the time step as

$$\mathbf{F}_p^{s,Etr} = \left(\mathbf{I} + \Delta t \sum_{\mathbf{i}} \tilde{\mathbf{v}}_{\mathbf{i}}^{s,n+1} \nabla w_{\mathbf{i}p}\right) \mathbf{F}_p^{E,n}$$

and then finally projected to $\mathbf{F}_p^{E,sn+1}$ according to plastic flow using the details in Section 3.6.3.

3.6.2 Grid Momentum Update

We discretize Equation (3.7) in a variational MPM manner to derive the grid momentum and velocity update. Multiplying Equation (3.7) by an arbitrary function $\mathbf{u}(\mathbf{x}) = \sum_i \mathbf{u}_i N_i(\mathbf{x})$, taking an arbitrary $\mathbf{u}_i$, using $\rho\mathbf{v} = \rho\mathbf{v}^s + \mathbf{q}^w$ and treating the $\frac{D^s \rho \mathbf{v}^s}{Dt}$ term as in Equation (3.26), we can conclude

$$\int_{B^s_{t^n}} \left(\frac{\tilde{\rho}\tilde{\mathbf{v}}^s + \tilde{\mathbf{q}}^w - \rho\mathbf{v}^s - \mathbf{q}^w}{\Delta t} + (\rho\mathbf{v}^s + \mathbf{q}^w)\nabla \cdot \mathbf{v}^s \right) N_i d\mathbf{x} =$$
$$\int_{B^s_{t^n}} \mathbf{f} N_i - (\boldsymbol{\sigma} - \mathbf{v}^w \otimes \mathbf{q}^w) \nabla N_i d\mathbf{x} + \int_{\partial B^s_{t^n}} (\mathbf{t} + \beta^w \theta s \mathbf{v}^w) N_i ds(\mathbf{x})$$

with the convention that function $\tilde{f}$ is inherited as a function over $B^s_{t^n}$ as

$$\tilde{f} = f(\boldsymbol{\phi}^s({\boldsymbol{\phi}^s}^{-1}(\mathbf{x}, t^n), t^{n+1}), t^{n+1})$$

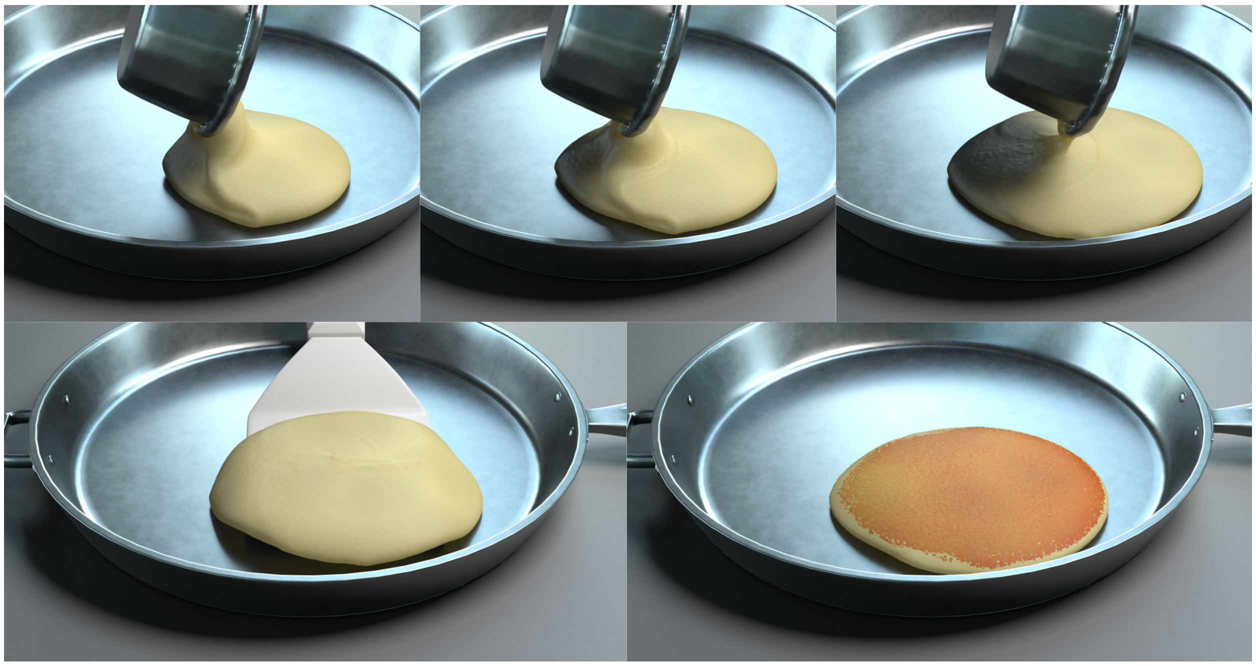


Figure 3.11: **Pancake**. Batter is poured into the pan (**top row**) and the pancake is flipped after the bottom is cooked(**bottom row**). The coloring is based on temperature.

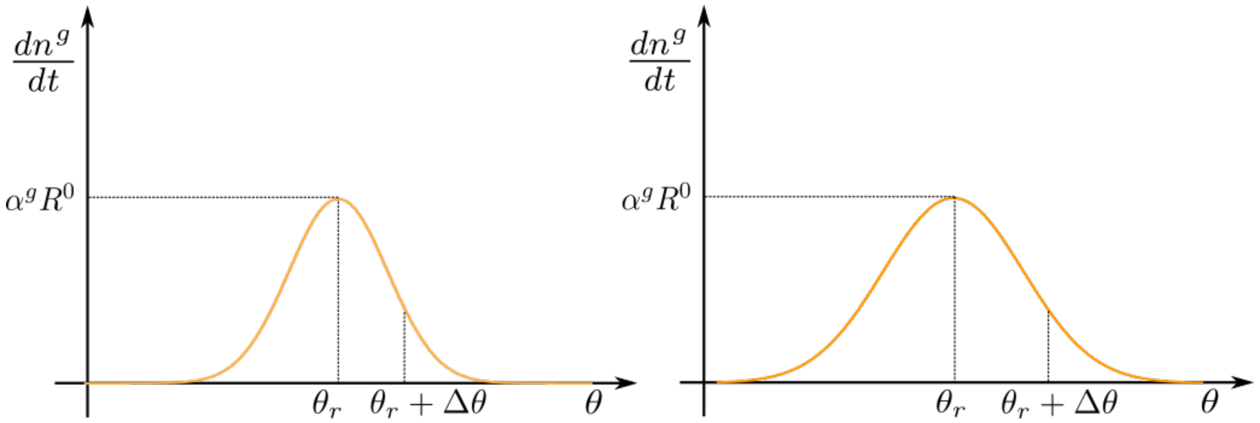


Figure 3.12: CO_2 **creation rate**. The carbon dioxide creation rate is a bell-shaped curve with the peak at θ_r. Bigger $\Delta\theta$ results in a flatter curve (**right**).

for functions f defined over $B^s_{t^{n+1}}$.

Using the particle positions $\mathbf{x}^n_p$ as the quadrature points with weights $V^{s,n}_p = J^{s,n}_p V^{s,0}_p$ we conclude

$$\sum_p \tilde{\rho}^{n+1}_p \tilde{\mathbf{v}}^{s,n+1}_p w_{\mathbf{i}p} V^{s,n}_p + \tilde{\mathbf{q}}^{w,n+1}_p w_{\mathbf{i}p} V^{s,n}_p = \tag{3.24}$$

$$\sum_p \rho^n_p \mathbf{v}^{s,n}_p w_{\mathbf{i}p} V^{s,n}_p + \mathbf{q}^{w,n}_p w_{\mathbf{i}p} V^{s,n}_p + \Delta t \Delta(\rho \mathbf{v}^s)_{\mathbf{i}} \tag{3.25}$$

$$\begin{aligned}
\Delta(\rho \mathbf{v}^s)_{\mathbf{i}} &= -\sum_p \left(\rho^n_p \mathbf{v}^{s,n}_p + \mathbf{q}^{w,n}_p\right) w_{\mathbf{i}p} V^{s,n}_p \sum_{\mathbf{k}} \mathbf{v}^{s,n}_{\mathbf{k}} \cdot \nabla w_{\mathbf{k}p} \\
&+ \sum_p \mathbf{f}^n_p w_{\mathbf{i}p} - \left(\boldsymbol{\sigma}^n_p - \mathbf{v}^{w,n}_p \otimes \mathbf{v}^{s,n}_p\right) V^{s,n}_p \nabla w_{\mathbf{i}p} \\
&+ \sum_{p \in \partial B^s_{t^n}} S_p w_{\mathbf{i}p} \left(\mathbf{t}^n_p + \beta^w \theta^n_p s^n_p \mathbf{v}^{w,n}_p\right)
\end{aligned}$$

where $\tilde{\rho}^{n+1}_p = \tilde{\rho}(\mathbf{x}^n_p, t^{n+1})$, $\rho^n_p = \rho(\mathbf{x}^n_p, t^n) = s^n_p \rho^w_0 + \frac{m^s_p}{V^{s,n}_p}$, and $\tilde{\mathbf{q}}^{w,n+1}_p = \tilde{\mathbf{q}}^w(\mathbf{x}^n_p, t^{n+1})$ is from Equation (3.23). We note that Equation (3.23) does not have a tilde superscript because it will be used with particle positions $\mathbf{x}^{n+1}_p$ at the end of the time step, whereas $\tilde{\mathbf{q}}^w(\mathbf{x}^n_p, t^{n+1})$ is the same value, but being used with time t^n positions $\mathbf{x}^n_p$. Noting that $\sum_p \rho^n_p \mathbf{v}^{s,n}_p w_{\mathbf{i}p} V^{s,n}_p \approx m^n_{\mathbf{i}} \mathbf{v}^{s,n}_{\mathbf{i}}$, we use Equation (3.24) as an update for the solid grid velocity as

$$\tilde{\mathbf{v}}^{s,n+1}_{\mathbf{i}} = \frac{1}{\tilde{m}^{n+1}_{\mathbf{i}}} \left(m^n_{\mathbf{i}} \mathbf{v}^{s,n}_{\mathbf{i}} + \Delta t \Delta(\rho \mathbf{v}^s)_{\mathbf{i}} + \sum_p \Delta \mathbf{q}_p w_{\mathbf{i}p} V^{s,n}_p \right),$$

where $\Delta \mathbf{q}_p = \mathbf{q}^{w,n}_p - \tilde{\mathbf{q}}^{w,n+1}_p$. We note that $\tilde{m}^{n+1}_{\mathbf{i}} = \tilde{m}^{w,n+1}_{\mathbf{i}} + m^{s,n}_{\mathbf{i}}$ where $\tilde{m}^{w,n+1}_{\mathbf{i}}$ is from Equation (3.28) and $m^{s,n}_{\mathbf{i}}$ is from Equation (3.16). We note that $m^{s,n}_{\mathbf{i}}$ does not change with time in this update since we track the motion of the solid constituents in a Lagrangian manner. Lastly we note that $\boldsymbol{\sigma}^n_p$ is a sum of the stresses in Equations (3.10), (3.14) and (3.15).

3.6.3 Return Mapping

Plasticity is applied by first assuming all deformation is elastic and getting a trial stress $\boldsymbol{\tau}^{s,Etr} = \frac{\partial\psi}{\partial \mathbf{F}^{s,E}}(\mathbf{F}_p^{s,Etr})(\mathbf{F}_p^{s,Etr})^{-T}$ from Equation (3.10). Following [SM93] we can write the return mapping discretization of Equations (3.12) and (3.13) in the form of a constrained minimization problem:

$$\boldsymbol{\tau}^{s,En+1} = \mathrm{argmin}_{\boldsymbol{\tau}\in\mathbb{E}}\frac{1}{2}(\boldsymbol{\tau}^{s,Etr} - \boldsymbol{\tau})^T\mathbf{C}^{-1}(\boldsymbol{\tau}^{s,Etr} - \boldsymbol{\tau})$$

where $\mathbb{E}$ stands for the elastic region bounded by the yield surface with $f(\boldsymbol{\tau}) \leq 0$, and $\mathbf{C}$ is the elastic tensor from $\boldsymbol{\tau}^{s,E} = \mathbf{C}\boldsymbol{\epsilon}$. For viscoplasticity the hard constraint that the minimization be over is $\mathbb{E}$ is replaced by a penalty term from viscosity and the unconstrained minimization

$$\boldsymbol{\tau}^{s,En+1} = \mathrm{argmin}\frac{1}{2}(\boldsymbol{\tau}^{s,Etr} - \boldsymbol{\tau})^T\mathbf{C}^{-1}(\boldsymbol{\tau}^{s,Etr} - \boldsymbol{\tau}) + \frac{\Delta t}{\eta}g(f(\boldsymbol{\tau}))$$

where Δt is the time step. To solve the minimization, we take the derivative with respect to $\boldsymbol{\tau}$ and set it equal to zero. The elastic Hencky strain can then be obtained trivially as $\boldsymbol{\epsilon}_p^{s,En+1} = \mathbf{C}^{-1}\boldsymbol{\tau}^{s,En+1}$. Then the new deformation gradient can be obtained using $\mathbf{F}_p^{s,En+1} = \mathbf{U}_p^{s,En+1}\exp\left(\boldsymbol{\Lambda}_p^{s,En+1}\right)\left(\mathbf{V}_p^{s,En+1}\right)^T$, where $\boldsymbol{\Lambda}_p^{s,En+1}$ is derived from the eigendecomposition $\boldsymbol{\epsilon}_p^{s,En+1} = \mathbf{U}_p^{s,En+1}\boldsymbol{\Lambda}_p^{s,En+1}\left(\mathbf{U}_p^{s,En+1}\right)^T$, and $\mathbf{U}_p^{s,En+1}$ and $\mathbf{V}_p^{s,En+1}$ are derived from the singular value decomposition $\mathbf{F}_p^{s,Etr} = \mathbf{U}_p^{s,En+1}\boldsymbol{\Sigma}_p^{s,Etr}\left(\mathbf{V}_p^{s,En+1}\right)^T$ respectively. This can be solved efficiently in terms of the eigenvalues of $\boldsymbol{\tau}^{s,En+1}$ and is similar to the methods in [KGP16, GGT18]. More details of the derivation is given in Section 3.9.1 and we illustrate the return mapping in the zero porosity case in Figure 3.14.

3.6.4 Water Mass Update

Our discretization is based on the weak form of Equation (3.4). Given an arbitrary function $u : B_{tn}^s \to \mathbb{R}$ we can conclude

$$\begin{aligned} 0 &= \int_{B_{tn}^s} \frac{D^s\rho^w}{Dt}u + \rho^w\nabla\cdot\mathbf{v}^s u + u\nabla\cdot\mathbf{q}^w d\mathbf{x} \\ &= \int_{B_{tn}^s} \frac{D^s\rho^w}{Dt}u + \rho^w\nabla\cdot\mathbf{v}^s u - \nabla u\cdot\mathbf{q}^w d\mathbf{x} + \int_{\partial B_{tn}^s} u\beta^w\theta s ds(\mathbf{x}) \end{aligned}$$

by integrating by parts and using the water flux boundary condition in Section 3.3.2. Furthermore, we note that

$$\begin{aligned}
&\int_{B^s_{t^n}} \frac{D^s\rho^w}{Dt}(\mathbf{x},t)u(\mathbf{x})d\mathbf{x} \qquad (3.26)\\
&= \int_{B^s_0} \frac{d}{dt}\rho^w(\boldsymbol{\phi}^s(\mathbf{X},t^n),t^n)u(\boldsymbol{\phi}^s(\mathbf{X},t^n))J^s(\mathbf{X},t^n)d\mathbf{X}\\
&\approx \int_{B^s_0} \frac{\rho^w(\boldsymbol{\phi}^s(\mathbf{X},t^{n+1}),t^{n+1}) - \rho^w(\boldsymbol{\phi}^s(\mathbf{X},t^n),t^n)}{\Delta t}u(\boldsymbol{\phi}^s(\mathbf{X},t^n))J^s(\mathbf{X},t^n)d\mathbf{X}\\
&= \int_{B^s_{t^n}} \frac{\tilde{\rho}^w(\mathbf{x},t^{n+1}) - \rho^w(\mathbf{x},t^n)}{\Delta t}u(\mathbf{x})d\mathbf{x}
\end{aligned}$$

where $\tilde{\rho}^w(\mathbf{x},t^{n+1}) = \rho^w(\boldsymbol{\phi}^s(\boldsymbol{\phi}^{s^{-1}}(\mathbf{x},t^n),t^{n+1}),t^{n+1})$ and $\boldsymbol{\phi}^{s^{-1}}(\mathbf{x},t^n)$ is the inverse flow map of the solid constituents. Intuitively, $\tilde{\rho}(\mathbf{x},t^{n+1})$ is the time t^{n+1} mass density but pulled back to the time t^n spatial configuration of the solid constituents.

Letting $u(\mathbf{x}) = \sum_{\mathbf{j}} u_{\mathbf{j}}N_{\mathbf{j}}(\mathbf{x})$ be defined from interpolation over the grid, we can then conclude

$$\begin{aligned}
0 &= \int_{B^s_{t^n}} \left(\frac{D^s\rho^w}{Dt} + \rho^w\nabla\cdot\mathbf{v}^s\right)\sum_{\mathbf{j}} u_{\mathbf{j}}N_{\mathbf{j}} - \sum_{\mathbf{j}} u_{\mathbf{j}}\nabla N_{\mathbf{j}}\cdot\mathbf{q}^w d\mathbf{x}\\
&+ \int_{\partial B^s_{t^n}} \sum_{\mathbf{j}} u_{\mathbf{j}}N_{\mathbf{j}}\beta^w\theta s ds(\mathbf{x})\\
&\approx \int_{B^s_{t^n}} \left(\frac{\tilde{\rho}^w - \rho^w}{\Delta t} + \rho^w\nabla\cdot\mathbf{v}^s\right)\sum_{\mathbf{j}} u_{\mathbf{j}}N_{\mathbf{j}} - \sum_{\mathbf{j}} u_{\mathbf{j}}\nabla N_{\mathbf{j}}\cdot\mathbf{q}^w d\mathbf{x}\\
&+ \int_{\partial B^s_{t^n}} \sum_{\mathbf{j}} u_{\mathbf{j}}N_{\mathbf{j}}\beta^w\theta s ds(\mathbf{x})
\end{aligned}$$

If we use the positions $\mathbf{x}^n_p$ as quadrature points with weights $V^{s,n}_p$ for volume integrals and $S^{s,n}_p$ for surface integrals, then we can further approximate as

$$\begin{aligned}
0 &= \frac{1}{\Delta t}\sum_p(\tilde{\rho}^{w,n+1}_p - \rho^{w,n}_p)V^{s,n}_p\sum_{\mathbf{j}} u_{\mathbf{j}}w_{\mathbf{j}p} +\\
&\sum_p \rho^{w,n}_p V^{s,n}_p\nabla\cdot\mathbf{v}^s(\mathbf{x}^n_p,t^n)\sum_{\mathbf{j}} u_{\mathbf{j}}w_{\mathbf{j}p} - \sum_p V^{s,n}_p\mathbf{q}^{w,n}_p\cdot\sum_{\mathbf{j}} u_{\mathbf{j}}\nabla w_{\mathbf{j}p}\\
&+ \sum_{p\in\partial B^s_{t^n}} \beta^w\theta^n_p s^n_p S^{s,n}_p\sum_{\mathbf{j}} u_{\mathbf{j}}w_{\mathbf{j}p}
\end{aligned}$$

where $\tilde{\rho}_p^{w,n+1} = \tilde{\rho}(\mathbf{x}_p^n, t^{n+1})$ and $\rho_p^{w,n} = \rho(\mathbf{x}_p^n, t^n)$. Since this must hold for any $u_{\mathbf{j}}$, we choose $u_{\mathbf{j}} = \delta_{\mathbf{ij}}$ to conclude

$$
\begin{aligned}
0 = &\sum_p \frac{(\tilde{\rho}_p^{w,n+1} - \rho_p^{w,n})}{\Delta t} V_p^{s,n} w_{\mathbf{i}p} + \sum_p \rho_p^{w,n} V_p^{s,n} \nabla \cdot \mathbf{v}^s(\mathbf{x}_p^n, t^n) w_{\mathbf{i}p} \\
&- \sum_p V_p^{s,n} \mathbf{q}_p^{w,n} \cdot \nabla w_{\mathbf{i}p} + \sum_{p \in \partial B_{t^n}^s} \beta^w \theta_p^n s_p^n S_p^{s,n} w_{\mathbf{i}p}
\end{aligned}
$$

for each node $\mathbf{i}$. Noting that $\rho_p^{w,n} V_p^{s,n} = m_p^{w,n}$, we can see that $m_{\mathbf{i}}^{w,n} = \sum_p \rho_p^{w,n} V_p^{s,n} w_{\mathbf{i}p}$ from Equation (3.16). With this observation, we can see that the discrete equations give us the following update of the grid water mass

$$
\tilde{m}_{\mathbf{i}}^{w,n+1} = m_{\mathbf{i}}^{w,n} + \Delta m_{\mathbf{i}}^{w,n} \tag{3.27}
$$

$$
\begin{aligned}
\Delta m_{\mathbf{i}}^{w,n} = &-\Delta t \sum_p m_p^{w,n} \sum_{\mathbf{k}} \mathbf{v}_{\mathbf{k}}^{s,n} \cdot \nabla w_{\mathbf{k}p} w_{\mathbf{i}p} \\
&+ \Delta t \sum_p V_p^{s,n} \mathbf{q}_p^{w,n} \cdot \nabla w_{\mathbf{i}p} - \Delta t \sum_{p \in \partial B_{t^n}^s} \beta^w \theta_p^n s_p^n S_p^{s,n} w_{\mathbf{i}p}
\end{aligned} \tag{3.28}
$$

where we use $\nabla \cdot \mathbf{v}^s(\mathbf{x}_p^n, t^n) = \sum_{\mathbf{k}} \mathbf{v}_{\mathbf{k}}^{s,n} \cdot \nabla w_{\mathbf{k}p}$ with $\mathbf{v}_{\mathbf{k}}^{s,n}$ from Equation (3.17). Here $\tilde{m}_{\mathbf{i}}^{w,n+1} = \sum_p \tilde{\rho}_p^{w,n+1} w_{\mathbf{i}p} V_p^{s,n}$ is the updated grid mass computed from the time t^{n+1} density and time t^n grid.

We use $\tilde{m}_{\mathbf{i}}^{w,n+1}$ to compute time t^{n+1} grid node saturation from $s_{\mathbf{i}}^{n+1} = \frac{\tilde{m}_{\mathbf{i}}^{w,n+1}}{\rho_0^w V_{\mathbf{i}}^{s,n}}$ where $V_{\mathbf{i}}^{s,n}$ is the representative volume of the grid node from Equation (3.19). This follows since $\tilde{m}_{\mathbf{i}}^{w,n+1} \approx \rho_{\mathbf{i}}^{w,n+1} V_{\mathbf{i}}^{s,n}$ and $\rho_{\mathbf{i}}^{w,n+1} = s_{\mathbf{i}}^{n+1} \rho_0^w$ where ρ_0^w is the initial spatially constant mass density of water.

3.6.5 Heat Transfer Update

The discretization of the heat energy balance equation is similar to that of the water mass update and [SSJ14]. We start with the Lagrangian form stated as Equation 1.10. For any test function $U(\mathbf{X}, 0) : B_0^s \to \mathbb{R}$, we have

$$
\int_{B_0^s} R_0 \alpha \frac{\partial \Theta}{\partial t} U d\mathbf{X} = \int_{B_0^s} \nabla \cdot (\mathbf{K} \nabla \Theta) U + \Theta \frac{\partial^2 \Psi}{\partial \mathbf{F} \partial \Theta} : \frac{d\mathbf{F}}{dt} U d\mathbf{X} \tag{3.29}
$$

Discretizing the left hand side with respect to time and space in the usual MPM style, and interpolating $U = \sum_{\mathbf{j}} U_{\mathbf{j}}(t) N_{\mathbf{j}}(\mathbf{X})$, $\Theta = \sum_{\mathbf{i}} \Theta_{\mathbf{i}}(t) N_{\mathbf{i}}(\mathbf{X})$ using shape functions $N_{\mathbf{j}}(\mathbf{X})$, we get

$$\begin{aligned}
\int_{B_0^s} R_0 \alpha \frac{\partial \Theta}{\partial t} U d\mathbf{X} &= \int_{B_0^s} R_0 \alpha \sum_{\mathbf{i}} \frac{\partial \Theta_{\mathbf{i}}}{\partial t} N_{\mathbf{i}} \sum_{\mathbf{j}} U_{\mathbf{j}} N_{\mathbf{j}} d\mathbf{X} \\
&\approx \int_{B_0^s} R(\mathbf{X}, t^n) J \alpha \sum_{\mathbf{i}} \frac{\tilde{\Theta}_{\mathbf{i}}^{n+1} - \Theta_{\mathbf{i}}^n}{\Delta t} N_{\mathbf{i}} \sum_{\mathbf{j}} U_{\mathbf{j}} N_{\mathbf{j}} d\mathbf{X} \\
&\approx \int_{B_{t^n}^s} \rho(\mathbf{x}, t^n) \alpha \sum_{\mathbf{i}} \frac{\tilde{\theta}_{\mathbf{i}}^{n+1} - \theta_{\mathbf{i}}^n}{\Delta t} N_{\mathbf{i}} \sum_{\mathbf{j}} u_{\mathbf{j}} N_{\mathbf{j}} d\mathbf{x} \\
&= \frac{1}{\Delta t} \sum_{\mathbf{i},\mathbf{j}} (\tilde{\theta}_{\mathbf{i}}^{n+1} - \theta_{\mathbf{i}}^n) u_{\mathbf{j}} \int_{B_{t^n}^s} \rho \alpha N_{\mathbf{i}} N_{\mathbf{j}} d\mathbf{x} \\
&\approx \frac{1}{\Delta t} \sum_{\mathbf{j}} \left[\alpha m_{\mathbf{j}}^n \tilde{\theta}_{\mathbf{j}}^{n+1} - \alpha m_{\mathbf{j}}^n \theta_{\mathbf{j}}^n \right] u_{\mathbf{j}}
\end{aligned}$$

Here $\tilde{\theta}^{n+1}$ stands for the time t^{n+1} temperature but pulled back to the time t^n spatial configuration. In the last step we used the mass lumping strategy. For the right hand side again using $\frac{d\mathbf{F}}{dt}(\mathbf{X}, t) = \nabla \mathbf{v} \mathbf{F}$ and with our choice of constitutive model we have

$$\Theta \frac{\partial^2 \Psi}{\partial \mathbf{F} \partial \Theta} : \frac{d\mathbf{F}}{dt} = -\Theta c \lambda \nabla \cdot \mathbf{v}$$

The right hand side can then be rewritten as

$$\begin{aligned}
&\int_{B_0^s} U \nabla \cdot (\mathbf{K} \nabla \Theta) + \Theta \frac{\partial^2 \Psi}{\partial \mathbf{F} \partial \Theta} : \frac{d\mathbf{F}}{dt} U d\mathbf{X} \\
=& \int_{\partial B_0^s} \mathbf{K} \nabla \Theta U \cdot \mathbf{N} ds(\mathbf{X}) - \int_{B_0^s} (\mathbf{K} \nabla \Theta \cdot \nabla U + c \lambda \Theta \nabla \cdot \mathbf{v} U) d\mathbf{X} \\
=& -\int_{\partial B_0^s} U \mathbf{Q} \cdot \mathbf{N} ds(\mathbf{X}) - \int_{B_0^s} (\mathbf{K} \nabla \Theta \cdot \nabla U + c \lambda \Theta \nabla \cdot \mathbf{v} U) d\mathbf{X} \\
=& -\int_{\partial B_{t^n}^s} u \mathbf{q} \cdot \mathbf{n} ds(\mathbf{x}) - \int_{B_{t^n}^s} \left(\frac{1}{J} \mathbf{K} \mathbf{F}^T \nabla \theta \cdot \mathbf{F}^T \nabla u + \frac{1}{J} c \lambda \theta \nabla \cdot \mathbf{v} u \right) d\mathbf{x} \\
=& -\int_{\partial B_{t^n}^s} u \mathbf{q} \cdot \mathbf{n} ds(\mathbf{x}) + \int_{B_{t^n}^s} \left(\left(-\frac{1}{J} \mathbf{F} \mathbf{K} \mathbf{F}^T \nabla \theta \right) \cdot \nabla u - \frac{1}{J} c \lambda \theta \nabla \cdot \mathbf{v} u \right) d\mathbf{x} \\
=& -\int_{\partial B_{t^n}^s} u \mathbf{q} \cdot \mathbf{n} ds(\mathbf{x}) + \int_{B_{t^n}^s} \left(\mathbf{q} \cdot \nabla u - \frac{1}{J} c \lambda \theta \nabla \cdot \mathbf{v} u \right) d\mathbf{x}
\end{aligned}$$

Discretizing with respect to space using shape functions we get

$$-\int_{\partial B^s_{tn}} u\mathbf{q}\cdot\mathbf{n}ds(\mathbf{x})+\int_{B^s_{tn}}\mathbf{q}\cdot\nabla u d\mathbf{x}-\int_{B^s_{tn}}\frac{1}{J}c\lambda\theta\nabla\cdot\mathbf{v}ud\mathbf{x}$$
$$=-\int_{\partial B^s_{tn}}\sum_{\mathbf{j}}u_{\mathbf{j}}(t)N_{\mathbf{j}}(\mathbf{x})\beta(\theta-\theta_{out})ds(\mathbf{x})+\int_{B^s_{tn}}\mathbf{q}\cdot\sum_{\mathbf{j}}u_{\mathbf{j}}\nabla N_{\mathbf{j}}d\mathbf{x}-\int_{B^s_{tn}}\frac{1}{J}c\lambda\theta\nabla\cdot\mathbf{v}\sum_{\mathbf{j}}u_{\mathbf{j}}N_{\mathbf{j}}d\mathbf{x}$$
$$\approx-\sum_{\mathbf{j}}u_{\mathbf{j}}\sum_{p\in\partial B^s_{tn}}S_p\beta(\theta_p-\theta_{out})w_{\mathbf{j}p}+\sum_{\mathbf{j}}u_{\mathbf{j}}\sum_p V_p\mathbf{q}_p\cdot\nabla w_{\mathbf{j}p}-\sum_{\mathbf{j}}u_{\mathbf{j}}\sum_p\frac{1}{J_p}V_pc\lambda\theta_p\nabla\cdot\mathbf{v}_pw_{\mathbf{j}p}$$
$$=-\sum_{\mathbf{j}}u_{\mathbf{j}}\sum_{p\in\partial B^s_{tn}}S_p\beta(\theta_p-\theta_{out})w_{\mathbf{j}p}+\sum_{\mathbf{j}}u_{\mathbf{j}}\sum_p V_p(\mathbf{q}_p\cdot\nabla w_{\mathbf{j}p}-\frac{1}{J_p}c\lambda\theta_p\nabla\cdot\mathbf{v}_pw_{\mathbf{j}p})$$

where $w_{\mathbf{j}p}=N_{\mathbf{j}}(\mathbf{x}_p)$, and S_p being the surface area of the local region tracked with particle p. Therefore the discrete form of the heat equation is given as

$$\frac{1}{\Delta t}\sum_{\mathbf{j}}\left[\alpha m^n_{\mathbf{j}}\tilde{\theta}^{n+1}_{\mathbf{j}}-\alpha m^n_{\mathbf{j}}\theta^n_{\mathbf{j}}\right]u_{\mathbf{j}}$$
$$=-\sum_{\mathbf{j}}u_{\mathbf{j}}\sum_{p\in\partial B^s_{tn}}S_p\beta(\theta_p-\theta_{out})w_{\mathbf{j}p}+\sum_{\mathbf{j}}u_{\mathbf{j}}\sum_p V_p(\mathbf{q}_p\cdot\nabla w_{\mathbf{j}p}-\frac{1}{J_p}c\lambda\theta_p\nabla\cdot\mathbf{v}_pw_{\mathbf{j}p})$$

Taking $u_{\mathbf{j}}=\delta_{\mathbf{ij}}$, we get

$$\frac{1}{\Delta t}\left[\alpha m^{s,n}_{\mathbf{i}}\tilde{\theta}^{n+1}_{\mathbf{i}}-\alpha m^{s,n}_{\mathbf{i}}\theta^n_{\mathbf{i}}\right]=\sum_{p\in\partial B^s_{tn}}S_p\beta(\theta_p-\theta_{out})w_{\mathbf{i}p}+\sum_p V^{s,n}_p(\mathbf{q}_p\cdot\nabla w_{\mathbf{i}p}-\frac{1}{J^{s,n}_p}c\lambda\theta_p\nabla\cdot\mathbf{v}_pw_{\mathbf{i}p})$$

Further note that we interpolate $\mathbf{q}_p\approx-\kappa\nabla\theta_p=-\sum_{\mathbf{j}}\kappa_{\mathbf{j}}\theta_{\mathbf{j}}\nabla w_{\mathbf{j}p}$ and $\nabla\cdot\mathbf{v}_p\approx\sum_{\mathbf{k}}\mathbf{v}_{\mathbf{k}}\cdot\nabla w_{\mathbf{k}p}$, We can therefore summarize the discrete heat equation as

$$\frac{1}{\Delta t}\left(\alpha m^{s,n}_{\mathbf{i}}\theta^{n+1}_{\mathbf{i}}-\alpha m^{s,n}_{\mathbf{i}}\theta^n_{\mathbf{i}}\right)=\sum_{p\in\partial B^s_{tn}}S_p\beta(\theta^{n+\gamma}_p-\theta_{out})w_{\mathbf{i}p}$$
$$-\sum_p V^{s,n}_p\left(\sum_{\mathbf{j}}\kappa_{\mathbf{j}}\theta^{n+\gamma}_{\mathbf{j}}\nabla w_{\mathbf{j}p}\cdot\nabla w_{\mathbf{i}p}+\frac{c\lambda\theta_p}{J^{s,n}_p}\sum_{\mathbf{k}}\mathbf{v}^{s,n}_{\mathbf{k}}\cdot\nabla w_{\mathbf{k}p}w_{\mathbf{i}p}\right)$$

where $\gamma=0$ for forward Euler and $\gamma=1$ for backward Euler. We note that the last term expresses the effects of thermal expansion, which were not considered in [SSJ14].

3.6.6 Boundary Conditions

We need to treat discrete boundary integrals in various terms in Sections 3.6.2, 3.6.4 and 3.6.5. We define the set of boundary particles $p\in\partial B^s_{tn}$ as follows. First, we define a grid

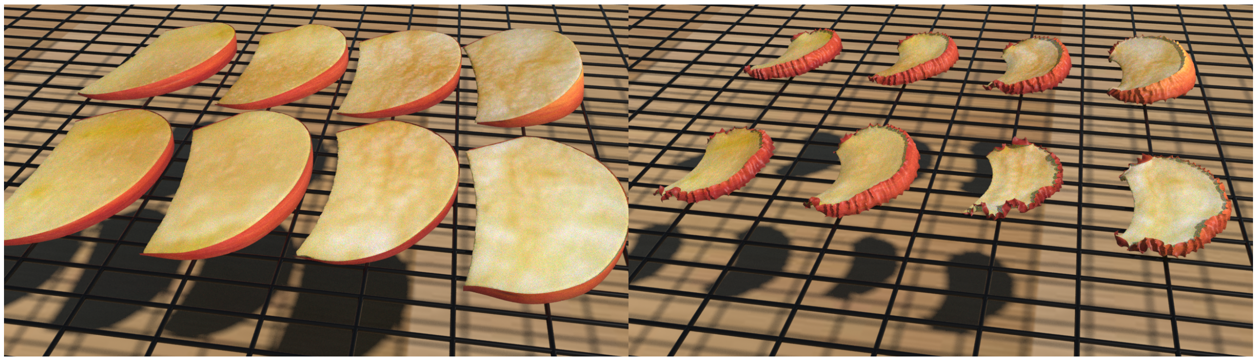


Figure 3.13: **Dehydrating apple slices**. Apple slices with various initial thickness are dried at constant temperature. The skin wrinkles and the slices buckle up and experience significant volume loss.

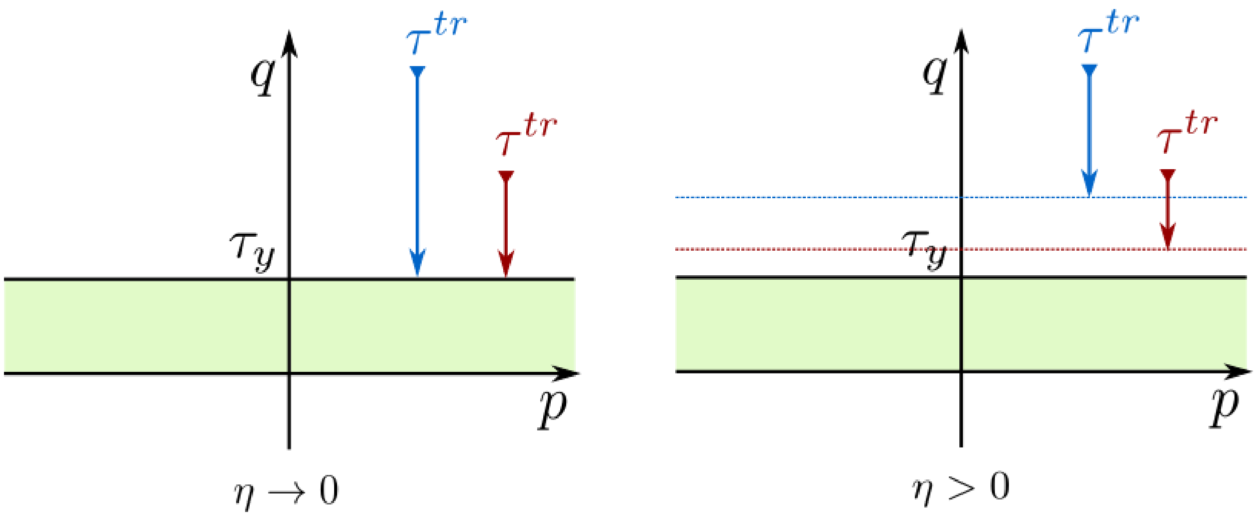


Figure 3.14: **Return mapping**. The projections of two trial state stresses are illustrated. With $\eta \to 0$ (**left**) the trial stresses are projected onto the yield surface; with $\eta > 0$ (**right**) the projections have the same directions but only a portion of the distance.

node $\mathbf{x}_i$ to be on the boundary if it has a neighboring grid node with zero mass m_i^n. Each boundary grid node then appends the particle closest to it to the set of boundary particles. We assume that each boundary particle owns a local spherical region, the volume of which is its representative volume, i.e. we have

$$V_p^{s,n} = \frac{4\pi}{3}(r_p^{s,n})^3$$

where $r_p^{s,n}$ is the approximated radius of the sphere. The surface area of the particle is then computed as $\pi(r_p^{s,n})^2$ which can be written in the form

$$S_p^{s,n} \approx (\frac{9\pi}{16}(V_p^{s,n})^2)^{1/3}$$

This serves as the quadrature weight at position $\mathbf{x}_p^n$ when needed for the surface integral approximation in Sections 3.6.2, 3.6.4 and 3.6.5.

3.7 Results

We demonstrate the efficacy of our model with several examples that illustrate the baking and cooking process of mixtures with various textures, and show our method is able to capture some of the key visual effects. Runtime performance is listed in Table 3.1. Simulations are run on an Intel Xeon E5-2687W v4 system with 48 threads and an Intel Xeon X5690 with 12 threads. We report the computation runtime in terms of average seconds per frame. Particle counts are given for each example. In general, we chose Δx so that there are approximately six particles per grid cell in the initial state. Δt is chosen in an adaptive manner restricted by a CFL condition that no particles are allowed to travel more than a portion of Δx in each time step.

3.7.1 Effect of Water Loss

We demonstrate the effect of the water content evolution with the simple scenario of fruit dehydration. In Figure 3.13 we simulate the process of drying apple slices on a rack. The flesh part is visco-elastoplastic with zero porosity and high initial saturation. Fruit skin contains much less moisture than the flesh and thus is modelled as a two-dimensional elastic sheet with zero water content. We use the Lagrangian energy approach of Jiang et al. [JSS15] to couple the elastic surface. Ambient temperature is set to be constant. The apple skin wrinkles and the slices twist and buckle out of the plane, matching real life observations. Figure 3.9 shows drying grapes with a similar setting. Water evaporates through the skin of the grapes and results in shrinkage. The skin again has a much lower water content and thus experiences less shrinkage, causing the many wrinkles and folds as it tries to retain its surface area while losing volume. Figure 3.10 visualizes the saturation change of the grapes, coloring from blue to green and finally red for decreasing saturation.

3.7.2 Temperature Dependent Plasticity

Figure 3.5 depicts a marshmallow roasting on an electric stove. The marshmallow is modeled as a homogeneous mixture in its initial state. Our model achieves the visual effect of an initial volume gain from the thermal expansion followed by a slightly burnt crust with a gooey center by using temperature dependent plasticity parameters. The coloration of the marshmallow is also rendered according to temperature. In Figure 3.15 we successfully capture the drastic difference of the behavior of the marshmallow when pressed between two crackers pre- and post-roasting, demonstrating our model's ability to track the significant texture changes of the mixture during the baking process. Figure 3.4 demonstrates baking of a lava cake in a ramekin. The initial batter is again a homogeneous mixture, and our model captures its transition to a cake with fully baked exterior and the characteristic molten center. We also show pouring liquid pancake batter into a frying pan followed by cooking to get a soft elastic pancake that can be flipped, see Figure 3.11. In Figure 3.3, we test baking the same cookie dough with varying oven settings. The three cookies on the bottom row from left to right are baked under decreasing temperatures. With the temperature being too high, the dough gets heated up very quickly, so the cookie does not get enough time to spread, and the dough baked under low temperature has the opposite issue. This comparison matches closely to real life observations. The temperature is also visualized in the right column. The color

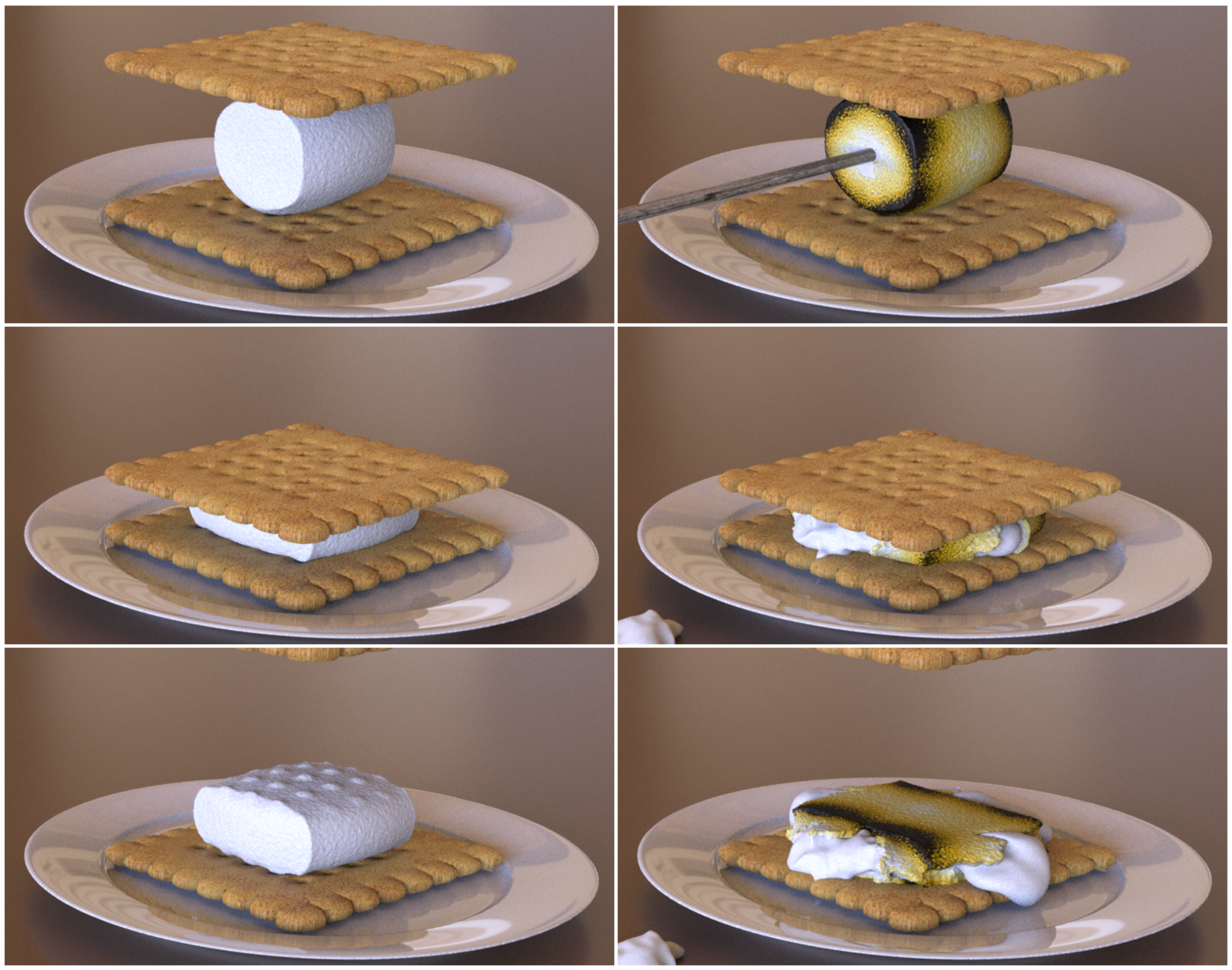


Figure 3.15: **Marshmallow compare**. The **left** column shows a marshmallow being squeezed down between two crackers, then springs back a little after the pressure is released, and the top cracker left markings on its surface. After roasting (**right** column) a crust forms on the exterior while the inside of the marshmallow is melted.

varies from blue to green then to red with increasing temperature.

3.7.3 Leavening

The influence of the chemical reaction aspect of our model is best illustrated by baking cookies with varying amount of leavening agents, as is shown in Figure 3.3. The top row of the cookies from left to right contains decreasing amount of leaveners and are baked under the same temperature. Too much leavening agent would produce a very tall cookie, while no leavener at all yields a flat one. The CO_2 creation and the corresponding pressure is the main contribution to the rising and expansion in our muffin (Figure 3.2), lava cake (Figure 3.4) and bread (Figure 3.16) examples.

3.7.4 Fracture

Our simulations are particle-based from their MPM conception, but for simulations with fracture, we construct a reference tetrahedron mesh in the initial state for rendering purposes and adopt the post-processing techniques from [WDG19] to obtain clean and consistent surfacing of the fractured material. The reference meshes are generated with TetWild [HZG18]. We demonstrate these effects with tearing examples in Figure 3.2 and Figure 3.17. By modeling the combined effect of water diffusion, temperature change and chemical leavening, our method is able to achieve visually realistic baking and tearing of a muffin, see Figure 3.2. Drawing slits on the bread dough helps with the rising during baking as well as the formation of a nice crust. In Figure 3.16 we compare the baking process of bread with and without scoring the surface beforehand. Notice how the bread cracks in a more controlled and appealing manner when there are slits on the surface.

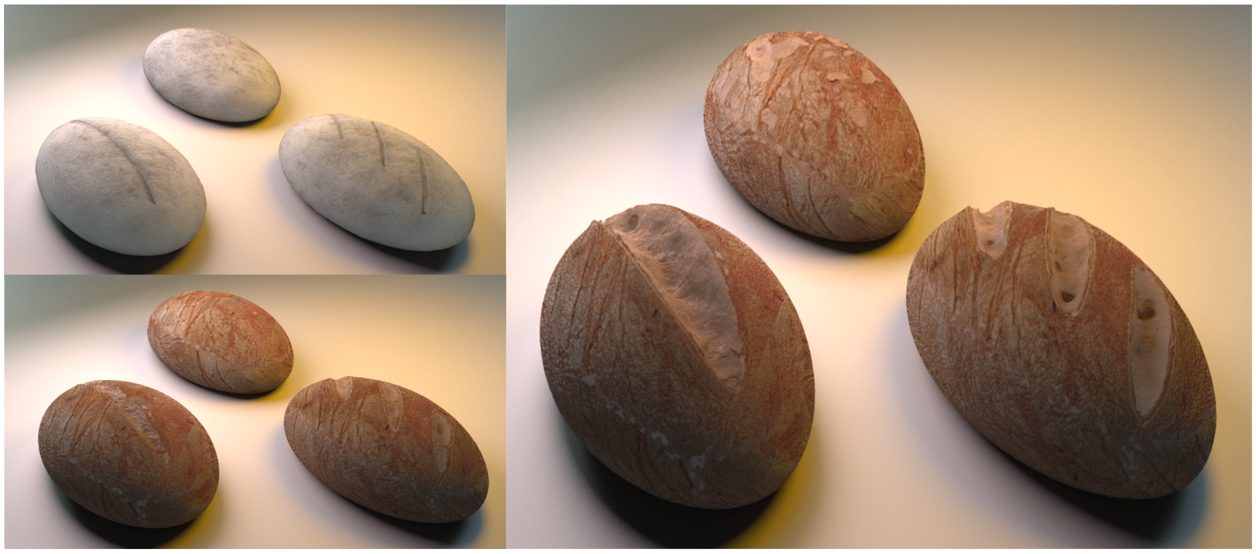


Figure 3.16: **Bread**. The **top left** shows raw dough, one is left intact and the other two have different slits on top. When baked(**right**) the breads expand in size and the slits open up. The bread without an initial slit also cracked on the top surface.

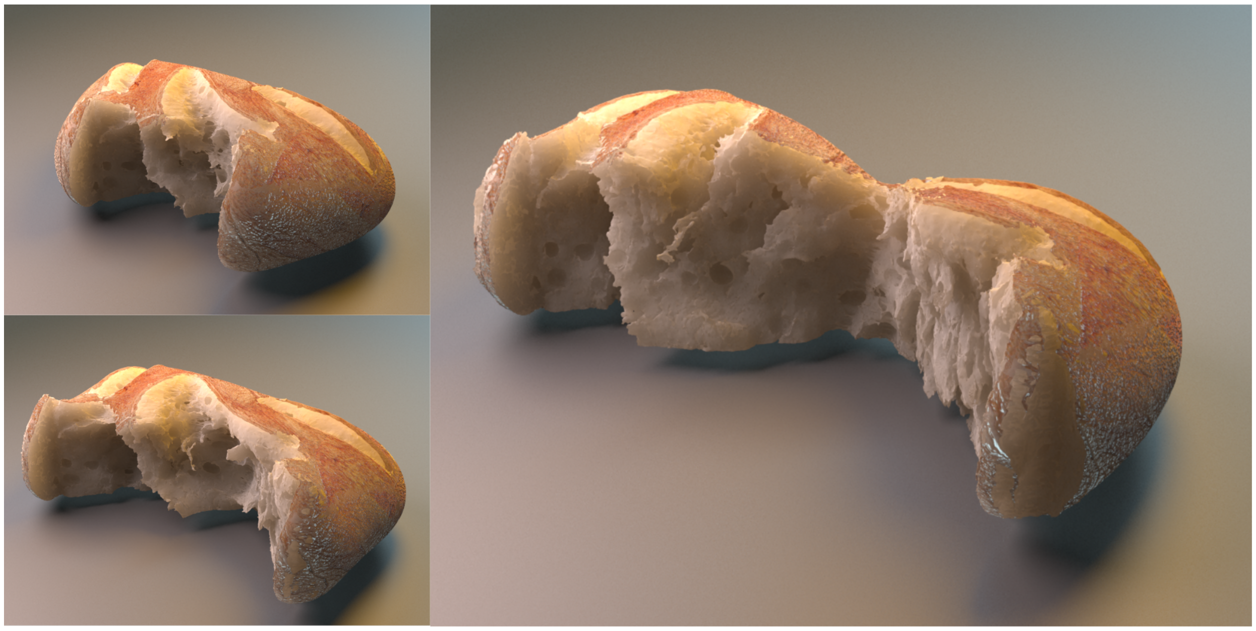


Figure 3.17: **Tearing bread**. We demonstrate tearing of the bread after baking. Surface meshes are generated through the mesh post-process of [WDG19] and are used for rendering.

3.8 Discussion and Limitations

We demonstrate that our method is capable of recreating a number of representative baking and cooking examples. However, our approach has a number of limitations. First, we simplify the mixture of non-water or gas constituents to be represented by a single phase. A more principled approach could be taken by including each of the flour, fat, egg, etc species in a mixture model. This would undoubtedly allow for a wider range of dough rheological behavior. Also, our kinematic assumption that the gas does not move relative to the solid mixture precludes diffusion of the gas through the dough. Lastly, due to the high-complexity of our coupled porous therm-mechanical model, we did not investigate fully implicit treatment of water diffusion and material stiffness. While we did not experience a need for excessively small time steps given the low stiffness and diffusion time scales in the materials considered, baked goods with faster water diffusion rates and material wave speeds could benefit from a fully implicit discretization.

3.9 Appendix

3.9.1 Viscoplasticity Return Mapping

In this section again we use **I** to denote the second order identity tensor (identity matrix), and $\mathcal{I}$ the fourth order identity tensor. For simplicity we denote our yield surface as

$$f(\boldsymbol{\tau}) = M^2p^2 + q^2 - M^2p_0^2 = 0$$

where M and p_0 are known values. The yield surface is an elliptical shape centered at the origin, with radius measures p_0 and p_0M, and so we can treat p_0 as a scaling factor on the size of the region of elastic response. The position of each point on the curve can then be represented under polar coordinates by its angle $\alpha \in [0, \pi]$ from the positive p-axis, and p_0. In this sense for any point $(\alpha, \bar{p})$ in the plane, we can think of $\frac{\bar{p}-p_0}{p_0}$ as the "distance" of the

Simulation	Time	Machine	Element	Particle
Bread bake (Fig 3.16)	20	1	3.2M	620K
Bread tear (Fig 3.17)	12	1	3.2M	620K
Cookies (per cookie) (Fig 3.3)	48	1	6.3M	1.2M
Pancake (Fig 3.11)	25	1	N/A	1M
Dough (per dough) (Fig 3.6)	11	1	2.7M	500K
Grape dehydration (Fig 3.9)	100	1	96K (surface)	346K
Muffin (Fig 3.2)	150	2	5.4M	1.2M
Apple dehydration (Fig 3.13)	25	2	76K (surface)	541K
Lava cake (Fig 3.4)	145	2	10.8M	2M
S'more (Fig 3.5)	130	2	N/A	1.1M

Table 3.1: All simulations were run on either Intel Xeon E5-2690 v4 system with 48 threads (Machine 1) or Intel Xeon X5690 with 12 threads (Machine 2). Simulation time is measured in average seconds per frame. Element denotes number of tetrahedra in the volumetric mesh or number of triangles in the surface mesh when applicable. Particle denotes the total number of MPM particles in the simulation.

point to the yield surface. More formally, we can define a canonical yield function $\tilde{f}$ as:

$$\tilde{f}(\alpha, \bar{p}) = \frac{\bar{p}}{p_0} - 1$$

Intuitively this is a signed distance function of the level set given by the yield surface. We use this notion as the yield criterion to simplify the computations in return mapping.

The return mapping is given by

$$\begin{aligned}
\boldsymbol{\tau}^{s,En+1} &= \text{argmin}_{\boldsymbol{\tau}} \frac{1}{2}(\boldsymbol{\tau}^{s,Etr} - \boldsymbol{\tau})^T \mathbf{C}^{-1}(\boldsymbol{\tau}^{s,Etr} - \boldsymbol{\tau}) + \frac{\Delta t}{\eta} g(f(\boldsymbol{\tau})) \\
&= \text{argmin}_{\boldsymbol{\tau}(\alpha,\bar{p})} \frac{1}{2}(\boldsymbol{\tau}^{s,Etr} - \boldsymbol{\tau})^T \mathbf{C}^{-1}(\boldsymbol{\tau}^{s,Etr} - \boldsymbol{\tau}) + \frac{\Delta t}{2\eta}(\frac{\bar{p}}{p_0} - 1)^2
\end{aligned}$$

We can write $\boldsymbol{\tau}, p, q$ in terms of α and $\bar{p}$:

$$\begin{aligned}
p &= \bar{p}\cos(\alpha) \\
q &= \bar{p} M \sin(\alpha) \\
\boldsymbol{\tau} &= -p\mathbf{I} + \sqrt{\frac{2}{3}} q \frac{\boldsymbol{\epsilon}^{dev}}{||\boldsymbol{\epsilon}^{dev}||}
\end{aligned}$$

where $\boldsymbol{\epsilon}^{dev} = \boldsymbol{\epsilon} - \frac{1}{3}tr(\boldsymbol{\epsilon})\mathbf{I}$ is the deviatoric part of $\boldsymbol{\epsilon} = \log(\boldsymbol{\Sigma})$. $\boldsymbol{\tau} = \mathcal{C}\boldsymbol{\epsilon}$ where $\mathcal{C} = 2\mu\mathcal{I} + \lambda\mathbf{I}\otimes\mathbf{I}$. We write the function to be minimized as $E(\alpha, \bar{p}) + \frac{\Delta t}{2\eta}(\frac{p}{p_0} - 1)^2$, and let $\mathcal{D} = \mathcal{C}^{-1}$, differentiating the above equation with respect to α and $\bar{p}$ we get

$$\begin{aligned}
0 = \frac{\partial E}{\partial \alpha} &= \frac{\partial E}{\partial \boldsymbol{\tau}} : (\frac{\partial \boldsymbol{\tau}}{\partial p}\frac{\partial p}{\partial \alpha} + \frac{\partial \boldsymbol{\tau}}{\partial q}\frac{\partial q}{\partial \alpha}) \\
&= \mathcal{D}(\boldsymbol{\tau}^{s,Etr} - \boldsymbol{\tau}) : (\bar{p}\sin(\alpha)\mathbf{I} + \sqrt{\frac{2}{3}}\frac{\boldsymbol{\epsilon}^{dev}}{||\boldsymbol{\epsilon}^{dev}||}\bar{p}M\cos(\alpha)) \\
&= (\boldsymbol{\epsilon}^{s,Etr} - \mathcal{D}\boldsymbol{\tau}) : (\bar{p}\sin(\alpha)\mathbf{I} + \sqrt{\frac{2}{3}}\frac{\boldsymbol{\epsilon}^{dev}}{||\boldsymbol{\epsilon}^{dev}||}\bar{p}M\cos(\alpha)) \\
0 = \frac{\partial E}{\partial \bar{p}} + \frac{\Delta t}{\eta}(\frac{\bar{p}}{p_0} - 1)\frac{1}{p_0} &= \frac{\partial E}{\partial \boldsymbol{\tau}} : (\frac{\partial \boldsymbol{\tau}}{\partial p}\frac{\partial p}{\partial \bar{p}} + \frac{\partial \boldsymbol{\tau}}{\partial q}\frac{\partial q}{\partial \bar{p}}) + \frac{\Delta t}{\eta}(\frac{\bar{p}}{p_0} - 1)\frac{1}{p_0} \\
&= (\boldsymbol{\epsilon}^{s,Etr} - \mathcal{D}\boldsymbol{\tau}) : (-\cos(\alpha)\mathbf{I} + M\sin\alpha\sqrt{\frac{2}{3}}\frac{\boldsymbol{\epsilon}^{dev}}{||\boldsymbol{\epsilon}^{dev}||}) + \frac{\Delta t}{\eta p_0}(\frac{\bar{p}}{p_0} - 1) \\
&= (\boldsymbol{\epsilon}^{s,Etr} - \mathcal{D}\boldsymbol{\tau}) : (-\cos(\alpha)\mathbf{I}) + \frac{\Delta t}{\eta p_0}(\frac{\bar{p}}{p_0} - 1)
\end{aligned}$$

Straight forward computation yields $\mathcal{D}\mathbf{I} = \frac{1}{2\mu+3\lambda}\mathbf{I}$ and $\mathcal{D}\boldsymbol{\epsilon}^{dev} = \frac{1}{2\mu}\boldsymbol{\epsilon}^{dev}$. Note that $\mathcal{D} = \mathcal{C}^{-1}$ is symmetric, we move it to the other side of the contraction, The equations above then becomes

$$\begin{aligned}
0 &= \boldsymbol{\epsilon}^{s,Etr} : (\bar{p}\sin(\alpha)\mathbf{I} + \sqrt{\frac{2}{3}}\frac{\boldsymbol{\epsilon}^{dev}}{||\boldsymbol{\epsilon}^{dev}||}\bar{p}M\cos(\alpha)) \\
&- \boldsymbol{\tau} : (\bar{p}\sin(\alpha)\frac{1}{2\mu+3\lambda}\mathbf{I} + \bar{p}M\cos(\alpha)\sqrt{\frac{2}{3}}\frac{\boldsymbol{\epsilon}^{dev}}{2\mu||\boldsymbol{\epsilon}^{dev}||}) \\
&= \bar{p}\sin(\alpha)tr(\boldsymbol{\epsilon}^{s,Etr}) + \sqrt{\frac{2}{3}}\frac{\bar{p}M\cos(\alpha)\boldsymbol{\epsilon} : \boldsymbol{\epsilon}^{dev}}{||\boldsymbol{\epsilon}^{dev}||} \\
&- \frac{\bar{p}\sin(\alpha)}{2\mu+3\lambda}tr(\boldsymbol{\tau}) - \sqrt{\frac{2}{3}}\frac{\bar{p}M\cos(\alpha)\boldsymbol{\tau} : \boldsymbol{\epsilon}^{dev}}{2\mu||\boldsymbol{\epsilon}^{dev}||} \\
0 &= \boldsymbol{\epsilon}^{s,Etr} : (-\cos(\alpha)\mathbf{I}) + \boldsymbol{\tau} : \frac{\cos(\alpha)}{2\mu+3\lambda}\mathbf{I} + \frac{\Delta t}{\eta p_0}(\frac{\bar{p}}{p_0} - 1) \\
&= -\cos(\alpha)tr(\boldsymbol{\epsilon}) + \frac{\cos(\alpha)}{2\mu+3\lambda}tr(\boldsymbol{\tau}) + \frac{\Delta t}{\eta p_0}(\frac{\bar{p}}{p_0} - 1)
\end{aligned}$$

We further have $tr(\boldsymbol{\tau}) = -3p = -3\bar{p}\cos(\alpha)$ and $\boldsymbol{\tau} : \boldsymbol{\epsilon}^{dev} = \sqrt{\frac{2}{3}}q||\boldsymbol{\epsilon}^{dev}|| = \sqrt{\frac{2}{3}}\bar{p}M\sin(\alpha)||\boldsymbol{\epsilon}^{dev}||$. Substitute the $\boldsymbol{\tau}$ terms with these expressions and grouping terms with respect to α and $\bar{p}$ we arrive at

$$\begin{aligned}
0 &= A\bar{p}\sin(\alpha) + B\cos(\alpha) + C\bar{p}^2\sin(\alpha)\cos(\alpha) + D\sin(\alpha)\cos(\alpha) \\
0 &= F\cos(\alpha) + G\bar{p}\cos^2(\alpha) + H\bar{p}
\end{aligned}$$

Where A, B, C, D, F, G, H are constant coefficients. Note that we can directly compute $\bar{p}$ from the second equation, plugging it back into the first equation we get $L(\alpha) = 0$ for some function L. We solve for α from this equation using Newton's method, and the corresponding $\bar{p}$ can be directly obtained. The projected stress $\boldsymbol{\tau}^{s,En+1}$ is then computed with the new α and $\bar{p}$.

References

[AAI12] G. Akinci, N. Akinci, M. Ihmsen, and M. Teschner. "An efficient surface reconstruction pipeline for particle-based fluids." In J. Bender, A. Kuijper, D. Fellner, and E. Guerin, editors, *Workshop on Virtual Reality Interaction and Physical Simulation*. The Eurographics Association, 2012.

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

[AIA12] G. Akinci, M. Ihmsen, N. Akinci, and M. Teschner. "Parallel surface reconstruction for particle-based fluids." *Comp Graph Forum*, **31**(6):1797–1809, 2012.

[And17] T. Anderson. *Fracture mechanics: fundamentals and applications*. CRC Press, 2017.

[APK07] B. Adams, M. Pauly, R. Keiser, and L. Guibas. "Adaptively sampled particle fluids." *ACM Trans Graph*, **26**(3), 2007.

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

[ATW13] R. Ando, N. Thürey, and C. Wojtan. "Highly adaptive liquid simulations on tetrahedral meshes." *ACM Trans Graph*, **32**(4):103:1–103:10, 2013.

[BB09] T. Brochu and R. Bridson. "Robust topological operations for dynamic explicit surfaces." *SIAM J Sci Comp*, **31**(4):2472–2493, 2009.

[BCX03] T. Belytschko, H. Chen, J. Xu, and G. Zi. "Dynamic crack propagation based on loss of hyperbolicity and a new discontinuous enrichment." *IntJ Num Meth Eng*, **58**(12):1873–1905, 2003.

[BDP17] M. Buler, P. Diehl, D. Pflger, S. Frey, F. Sadlo, T. Ertl, and M. Schweitzer. "Visualization of fracture progression in peridynamics." *Comp Graph*, **67**(C):45–57, 2017.

[BDW13] O. Busaryev, T. Dey, and H. Wang. "Adaptive fracture simulation of multi-layered thin plates." *ACM Trans Graph*, **32**(4):52:1–52:6, 2013.

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

[BGA17] H. Barreiro, I. García-Fernández, I. Alduán, and M.-A. Otaduy. "Conformation constraints for efficient viscoelastic fluid simulation." *ACM Transactions on Graphics (TOG)*, **36**(6):221, 2017.

[BGB15] H. Bhattacharya, Y. Gao, and A. Bargteil. “A level-set method for skinning animated particle data.” *IEEE Trans Vis Comp Graph*, **21**:315–327, 2015.

[BGO06] A. Bargteil, T. Goktekin, J. O’Brien, and J. Strain. “A semi-Lagrangian contouring method for fluid simulation.” *ACM Trans Graph*, **25**(1), 2006.

[Bio41] M. Biot. “General theory of three-dimensional consolidation.” *J App Phys*, **12**(2):155–164, 1941.

[BKR88] J. Brackbill, D. Kothe, and H. Ruppel. “FLIP: A low-dissipation, PIC method for fluid flow.” *Comp Phys Comm*, **48**:25–38, 1988.

[Bli82] J. Blinn. “A generalization of algebraic surface drawing.” *ACM Trans Graph*, **1**(3):235–256, 1982.

[BR86] J. Brackbill and H. Ruppel. “FLIP: A method for adaptively zoned, Particle-In-Cell calculations of fluid flows in two dimensions.” *J Comp Phys*, **65**:314–343, 1986.

[BS15] S. Bandara and K. Soga. “Coupling of soil deformation and pore fluid flow using material point method.” *Comp Geotech*, **63**:199–214, 2015.

[BT02] B. Broyart and G. Trystram. “Modelling heat and mass transfer during the continuous baking of biscuits.” *Journal of Food Engineering*, **51**(1):47–57, 2002.

[BWH07] A. Bargteil, C. Wojtan, J. Hodgins, and G. Turk. “A finite element method for animating large viscoplastic flow.” *ACM Trans Graph*, **26**(3), 2007.

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

[CC95] B. De Cindio and S. Correra. “Mathematical modelling of leavened cereal goods.” *Journal of Food Engineering*, **24**(3):379–403, 1995.

[Cho14] M. Choi. “Real-time simulation of ductile fracture with oriented particles.” *Comp Anim Virt Worlds*, **25**(3-4):457–465, 2014.

[CMI02] M. Carlson, P. J. Mucha, R. B. Van Horn III, and G. Turk. “Melting and flowing.” In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 167–174. ACM, 2002.

[CWX13] F. Chen, C. Wang, B. Xie, and H. Qin. “Flexible and rapid animation of brittle fracture using the smoothed particle hydrodynamics formulation.” *Comp Anim Virt Worlds*, **24**(3-4):215–224, 2013.

[CZZ18] W. Chen, F. Zhu, J. Zhao, S. Li, and G. Wang. “Peridynamics-based fracture animation for elastoplastic solids.” *Comp Graph Forum*, **37**(1):112–124, 2018.

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

[DBG14] F. Da, C. Batty, and E. Grinspun. "Multimaterial mesh-based surface tracking." *ACM Trans Graph*, **33**(4):112:1–112:11, 2014.

[DC98] M. Desbrun and M. Cani. "Active implicit surface for animation." In *Graph Int*, pp. 143–150, 1998.

[DGP17] F. Dagenais, J. Gagnon, and E. Paquette. "Detail-preserving explicit mesh projection and topology matching for particle-based fluids." *Comp Graph Forum*, **36**(8):444–457, 2017.

[DHW19] M. Ding, X. Han, S. Wang, T. Gast, and J. Teran. "A thermomechanical material point method for baking and cooking." *ACM Transactions on Graphics (TOG)*, **38**(6):1–14, 11 2019.

[DLC07] N. Daphalapurkar, H. Lu, D. Coker, and R. Komanduri. "Simulation of dynamic crack growth using the generalized interpolation material point (GIMP) method." *Int J Frac*, **143**(1):79–102, 2007.

[DLH10] F. Debaste, A. Léonard, V. Halloin, and B. Haut. "Microtomographic investigation of a yeast grain porous structure." *Journal of Food Engineering*, **97**(4):526–532, April 2010.

[FBG18] Y. Fei, C. Batty, E. Grinspun, and C. Zheng. "A multi-scale model for simulating liquid-fabric interactions." *ACM Trans Graph*, **37**(4):51:1–51:16, 2018.

[FF12] H. Faridi and J. Faubion. *Dough rheology and baked product texture*. Springer Science & Business Media, 2012.

[Fic55] A. Fick. "On liquid diffusion." *London, Edinburgh, and Dublin Philos Mag J Sci*, **10**:30–39, 1855.

[GBB09] D. Gerszewski, H. Bhattacharya, and A. Bargteil. "A point-based method for animating elastoplastic solids." In *Proc 2009 ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 133–138. ACM, 2009.

[GBG04] V. Guillard, B. Broyart, S. Guilbert, C. Bonazzi, and N. Gontard. "Moisture diffusivity and transfer modelling in dry biscuit." *Journal of Food Engineering*, **64**(1):81–87, 2004.

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

[GBT07] M. Gissler, M. Becker, and M. Teschner. "Constraint sets for topology-changing finite element models." In *VRIPHYS*, pp. 21–26, 2007.

[GGT18] J. Gaume, T. Gast, J. Teran, A. van Herwijnen, and C. Jiang. “Dynamic anticrack propagation in snow.” *Nature Com*, **9**(1):3047, 2018.

[GHF18] Q. Guo, X. Han, C. Fu, T. Gast, R. Tamstorf, and J. Teran. “A material point method for thin shells with frictional contact.” *ACM Trans Graph*, **37**(4):147, 2018.

[GN06] J. Guo and J. Nairn. “Three-Dimensional Dynamic Fracture Analysis Using the Material Point Method.” *Comp Mod Eng Sci*, **16**, 2006.

[GPH18] M. Gao, A. Pradhana, X. Han, Q. Guo, G. Kot, E. Sifakis, and C. Jiang. “Animating fluid sediment mixture in particle-laden flows.” *ACM Trans Graph*, **37**(4):149:1–149:11, 2018.

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

[GTJ17] M. Gao, A. Tampubolon, C. Jiang, and E. Sifakis. “An adaptive generalized interpolation material point method for simulating elastoplastic materials.” *ACM Trans Graph*, **36**(6):223:1–223:12, 2017.

[HA92] M. Haghi and L. Anand. “A constitutive model for isotropic, porous, elastic-viscoplastic metals.” *Mechanics of Materials*, **13**(1):37–53, March 1992.

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

[HZG18] Y. Hu, Q. Zhou, X. Gao, A. Jacobson, D. Zorin, and D. Panozzo. “Tetrahedral Meshing in the Wild.” *ACM Trans. Graph.*, **37**(4):60:1–60:14, July 2018.

[HZM11] P. Huang, X. Zhang, S. Ma, and X. Huang. “Contact algorithms for the material point method in impact and penetration simulation.” *Int J Num Meth Eng*, **85**(4):498–517, 2011.

[JGT17] C. Jiang, T. Gast, and J. Teran. “Anisotropic elastoplasticity for cloth, knit and hair frictional contact.” *ACM Trans Graph*, **36**(4):152, 2017.

[JML16] B. Jones, A. Martin, J. Levine, T. Shinar, and A. Bargteil. “Ductile fracture for clustered shape matching.” In *Proc ACM SIGGRAPH Symp Int 3D Graph Games*, pp. 65–70. ACM, 2016.

[JSS15] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin. “The Affine Particle-In-Cell Method.” *ACM Trans Graph*, **34**(4):51:1–51:10, 2015.

[JSV13] I. Jassim, D. Stolle, and P. Vermeer. “Two-phase dynamic analysis by material point method.” *Int J Num Anal Meth Geomech*, **37**(15):2502–2522, 2013.

[KAG05] R. Keiser, B. Adams, D. Gasser, P. Bazzi, P. Dutré, and M. Gross. "A unified lagrangian approach to solid-fluid animation." In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.*, pp. 125–148. IEEE, 2005.

[KBT17] D. Koschier, J. Bender, and N. Thuerey. "Robust eXtended Finite Elements for complex cutting of deformables." *ACM Trans Graph*, **36**(4):55:1–55:13, 2017.

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

[KMB08] P. Kaufmann, S. Martin, M. Botsch, and M. Gross. "Flexible simulation of deformable models using discontinuous Galerkin FEM." In *Proc 2008 ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 105–115. Eurographics Association, 2008.

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

[LC87] W. Lorensen and H. Cline. "Marching cubes: A high resolution 3D surface construction algorithm." *SIGGRAPH Comput Graph*, **21**(4):163–169, 1987.

[LGS09] W. van der Laan, S. Green, and M. Sainz. "Screen space fluid rendering with curvature flow." In *Proc 2009 Symp Int 3D Graph Games*, I3D '09, pp. 91–98. ACM, 2009.

[LHL11] N. Liu, X. He, S. Li, and G. Wang. "Meshless simulation of brittle fracture." *Comp Anim Virt Worlds*, **22**(2-3):115–124, 2011.

[LIG06] F. Losasso, G. Irving, E. Guendelman, and R. Fedkiw. "Melting and burning solids into liquids and gases." *IEEE Transactions on Visualization and Computer Graphics*, **12**(3):343–352, 2006.

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

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

[LYC15] Y. Liu, X. Yang, Y. Cao, Z. Wang, B. Chen, J. Zhang, and H. Zhang. "Dehydration of core/shell fruits." *Computers & Graphics*, **47**:68–77, April 2015.

[M09] Matthias Müller. "Fast and robust tracking of fluid surfaces." In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 237–245. ACM, 2009.

[MBF04] N. Molino, Z. Bao, and R. Fedkiw. "A virtual node algorithm for changing mesh topology during simulation." *ACM Trans Graph*, **23**(3):385–392, 2004.

[MBT15] O. Mercier, C. Beauchemin, N. Thuerey, T. Kim, and D. Nowrouzezahrai. "Surface turbulence for particle-based liquid simulations." *ACM Trans Graph*, **34(6)**:10, Nov 2015.

[MCG03] M. Müller, D. Charypar, and M. Gross. "Particle-based fluid simulation for interactive applications." In *Proc 2003 ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 154–159. Eurographics Association, 2003.

[MCK13] M. Müller, N. Chentanez, and T. Kim. "Real-time dynamic fracture with volumetric approximate convex decompositions." *ACM Trans Graph*, **32**(4):115:1–115:10, 2013.

[MCZ07] K. Museth, M. Clive, and B. Zafar. "Blobtacular: surfacing particle system in "Pirates of the Caribbean 3"." In *ACM SIGGRAPH 2007 Sketches*, SIGGRAPH '07. ACM, 2007.

[MG04] M. Müller and M. Gross. "Interactive virtual materials." In *Proc Graph Int*, pp. 239–246. Canadian Human-Computer Communications Society, 2004.

[MGG10] N. Maréchal, E. Guérin, E. Galin, S. Mérillou, and N. Mérillou. "Heat transfer simulation for modeling realistic winter sceneries." In *Computer Graphics Forum*, volume 29, pp. 449–458. Wiley Online Library, 2010.

[MHH07] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. "Position based dynamics." *J Vis Comm Im Rep*, **18**(2):109–118, 2007.

[MHT05] M. Müller, B. Heidelberger, M. Teschner, and M. Gross. "Meshless deformations based on shape matching." In *ACM transactions on graphics (TOG)*, volume 24, pp. 471–478. ACM, 2005.

[MKN04] M. Müller, R. Keiser, A. Nealen, M. Pauly, M. Gross, and M. Alexa. "Point based animation of elastic, plastic and melting objects." In *Proc ACM SIGGRAPH/Eurograp Symp Comp Anim*, pp. 141–151, 2004.

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

[MSD07] M. Müller, S. Schirm, and S. Duthaler. "Screen space meshes." In *Proc 2007 ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 9–15. Eurographics Association, 2007.

[Mus14] K. Museth. *A flexible image processing approach to the surfacing of particle-based fluid animation*, pp. 81–84. 2014.

[Nai03] John A. Nairn. "Material point method calculations with explicit cracks." 2003.

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

[NSG14] V. Nicolas, P. Salagnac, P. Glouannec, J.-P. Ploteau, V. Jury, and L. Boillereaux. "Modelling heat and mass transfer in deformable porous media: Application to bread baking." *Journal of Food Engineering*, **130**:23–35, June 2014.

[OBH02] J. O'Brien, A. Bargteil, and J. Hodgins. "Graphical modeling and animation of ductile fracture." In *Proc ACM SIGGRAPH 2002*, pp. 291–294, 2002.

[OH99] J. O'Brien and J. Hodgins. "Graphical modeling and animation of brittle fracture." In *Proc 26th Conf Comp Graph Int Tech*, SIGGRAPH '99, pp. 137–146. ACM Press/Addison-Wesley Publishing Co., 1999.

[OKN09] M. Ohta, Y. Kanamori, and T. Nishita. "Deformation and fracturing using adaptive shape matching with stiffness adjustment." *Comp Anim Virt Worlds*, **20**(2–3):365–373, 2009.

[PKA05] M. Pauly, R. Keiser, B. Adams, P. Dutré, M. Gross, and L. Guibas. "Meshless animation of fracturing solids." *ACM Trans Graph*, **24**(3):957–964, 2005.

[PKK03] M. Pauly, R. Keiser, L. Kobbelt, and M. Gross. "Shape modeling with point-sampled geometry." *ACM Trans Graph*, **22**(3):641–650, 2003.

[PNJ14] T. Pfaff, R. Narain, J. de Joya, and J. O'Brien. "Adaptive tearing and cracking of thin sheets." *ACM Trans Graph*, **33**(4):110:1–110:9, 2014.

[PO09] E. Parker and J. O'Brien. "Real-time deformation and fracture in a game environment." In *Proc ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 165–175. ACM, 2009.

[Rag02] S. Raghavachary. "Fracture generation on polygonal meshes using Voronoi polygons." In *ACM SIGGRAPH 2002 Conf Abstracts App*, SIGGRAPH '02, pp. 187–187. ACM, 2002.

[RB68] K. Roscoe and J. Burland. "On the generalized stress-strain behaviour of wet clay." *Eng Plast*, pp. 535–609, 1968.

[RB08] I. Rosenberg and K. Birdwell. "Real-time particle isosurface extraction." In *Proc 2008 Symp Int 3D Graph Games*, pp. 35–43. ACM, 2008.

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

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

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

[SB12] E. Sifakis and J. Barbic. "FEM simulation of 3D deformable solids: a practitioner's guide to theory, discretization and model reduction." In *ACM SIGGRAPH 2012 Courses*, SIGGRAPH '12, pp. 20:1–20:50, New York, NY, USA, 2012. ACM.

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

[SKI13] M. Sakin-Yilmazer, T. Kemerli, H. Isleroglu, O. Ozdestan, G. Guven, A. Uren, and F. Kaymak-Ertekin. "Porous media based model for deep-fat vacuum frying potato chips." *Journal of Food Engineering*, **110**(3):483–489, December 2013.

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

[SO14] S. Schvartzman and M. Otaduy. "Fracture animation based on high-dimensional voronoi diagrams." In *Proc ACM SIGGRAPH Symp Int 3D Graph Games*, pp. 15–22. ACM, 2014.

[SOG09] D. Steinemann, M. Otaduy, and M. Gross. "Splitting meshless deforming objects with explicit surface tracking." *Graph Models*, **71**(6):209–220, 2009.

[SS07] C. Shen and A. Shah. "Extracting and parametrizing temporally coherent surfaces from particles." In *SIGGRAPH Sketches*, p. 66, 2007.

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

[SSF09] J. Su, C. Schroeder, and R. Fedkiw. "Energy stability and fracture for frame rate rigid body simulations." In *Proc 2009 ACM SIGGRAPH/Eurograph Symp Comp Anim*, pp. 155–164. ACM, 2009.

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

[SSP07] B. Solenthaler, J. Schläfli, and R. Pajarola. "A unified particle model for Fluid-solid interactions." *Comp Anim Virt Worlds*, **18**(1):69–82, 2007.

[SZS95] D. Sulsky, S. Zhou, and H. Schreyer. "Application of a particle-in-cell method to solid mechanics." *Comp Phys Comm*, **87**(1):236–252, 1995.

[TF88] D. Terzopoulos and K. Fleischer. "Modeling inelastic deformation: viscolelasticity, plasticity, fracture." *SIGGRAPH Comp Graph*, **22**(4):269–278, 1988.

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

[TGK17] A. P. Tampubolon, T. Gast, G. Klár, C. Fu, J. Teran, C. Jiang, and K. Museth. "Multi-species simulation of porous sand and water mixtures." *ACM Trans Graph*, **36**(4), 2017.

[THM04] M. Teschner, B. Heidelberger, M. Muller, and M. Gross. "A versatile and robust model for geometrically complex deformable solids." In *Proceedings Computer Graphics International, 2004.*, pp. 312–319. IEEE, 2004.

[TPF91] D. Terzopoulosi, J. Platt, and K. Fleischer. "Heating and melting deformable models." *The Journal of Visualization and Computer Animation*, **2**(2):68–73, 1991.

[TS98] K. Thorvaldsson and C. Skjöldebrand. "Water diffusion in bread during baking." *LWT-Food Science and Technology*, **31**(7-8):658–663, 1998.

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

[VLT09] F. Vanin, T. Lucas, and G. Trystram. "Crust formation and its role during bread baking." *Trends in Food Science & Technology*, **20**(8):333–343, 2009.

[WAM17] J. Wretborn, R. Armiento, and K. Museth. "Animation of crack propagation by means of an extended multi-body solver for the material point method." *Comp Graph*, **69**(C):131–139, 2017.

[WBG07] M. Wicke, M. Botsch, and M. Gross. "A finite element method on convex polyhedra." *Comp Graph Forum*, **26**:355–364, 2007.

[WDG19] S. Wang, M. Ding, T. Gast, L. Zhu, S. Gagniere, C. Jiang, and J. Teran. "Simulation and Visualization of Ductile Fracture with the Material Point Method." volume 2, p. 18. ACM, 2019.

[Wil08] B. Williams. *Fluid surface reconstruction from particles*. PhD thesis, University of British Columbia, 2008.

[WLK03] X. Wei, W. Li, and A. Kaufman. "Melting and Flowing of Viscous Volumes." In *Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003)*, p. 54. IEEE Computer Society, 2003.

[WRK10] M. Wicke, D. Ritchie, B. Klingner, S. Burke, J. Shewchuk, and J. O'Brien. "Dynamic local remeshing for elastoplastic simulation." *ACM Trans Graph*, **29**(4):49:1–11, 2010.

[WT08] C. Wojtan and G. Turk. "Fast viscoelastic behavior with thin features." *ACM Trans Graph*, **27**(3):1–8, 2008.

[WTG04] M. Wicke, M. Teschner, and M. Gross. "CSG tree rendering for point-sampled objects." In *12th Pac Graph*, pp. 160–168, 2004.

[WTG09] C. Wojtan, N. Thürey, M. Gross, and G. Turk. "Deforming meshes that split and merge." *ACM Trans Graph*, **28**(3):76:1–76:10, 2009.

[WTG10] C. Wojtan, N. Thürey, M. Gross, and G. Turk. "Physics-inspired topology changes for thin fluid features." *ACM Trans Graph*, **29**(4):50:1–50:8, 2010.

[XZY18] Xiangyun Xiao, Shuai Zhang, and Xubo Yang. "Fast, high-quality rendering of liquids generated using large scale SPH simulation." *J Comp Graph Tech*, **7**(1):17–39, 2018.

[YAP15] A. Yerro, E. Alonso, and N. Pinyol. "The material point method for unsaturated soils." *Geotechnique*, **65**(3):201–217, 2015.

[YCL17] T. Yang, J. Chang, M. C. Lin, R. R. Martin, J. J. Zhang, and S.-M. Hu. "A unified particle system framework for multi-phase, multi-material visual simulations." *ACM Transactions on Graphics (TOG)*, **36**(6):224, 2017.

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

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

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

[YSC18] Y. Yue, B. Smith, P. Chen, M. Chantharayukhonthorn, K. Kamrin, and E. Grinspun. "Hybrid grains: adaptive coupling of discrete and continuum simulations of granular media." *ACM Trans Graph*, **37**(6):283:1–283:19, 2018.

[YT13] J. Yu and G. Turk. "Reconstructing surfaces of particle-based fluids using anisotropic kernels." *ACM Trans. Graph.*, **32**(1):5:1–5:12, 2013.

[YWT12] J. Yu, C. Wojtan, G. Turk, and C. Yap. "Explicit mesh surfaces for particle based fluids." *Comp Graph Forum*, **31**(2pt4):815–824, 2012.

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

[ZD06] J. Zhang and A. Datta. "Mathematical modeling of bread baking process." *Journal of Food Engineering*, **75**(1):78–89, July 2006.

[ZJ10] C. Zheng and D. James. "Rigid-body fracture sound with precomputed soundbanks." *ACM Trans Graph*, **29**(4):69:1–69:13, 2010.

[ZLL] G. Zhang, D. Lu, and H. Liu. "Visualization of fluid simulation: An SPH-based multi-resolution method." *Concurrency and Computation: Practice and Experience*, p. e4509.

[ZP13] Q. Zhao and P. Papadopoulos. "Modeling and simulation of liquid diffusion through a porous finitely elastic solid." *Computational Mechanics*, **52**(3):553–562, September 2013.

[ZWC09] H. Zhang, K. Wang, and Z. Chen. "Material point method for dynamic analysis of saturated porous media under external contact/impact of solid bodies." *Comp Meth App Mech Eng*, **198**(17):1456–1472, 2009.

[ZWQ06] Y. Zhao, L. Wang, F. Qiu, A. Kaufman, and K. Mueller. "Melting and flowing in multiphase environment." *Computers & Graphics*, **30**(4):519–528, 2006.

[ZZS06] N. Zhang, X. Zhou, D. Sha, X. Yuan, K. Tamma, and B. Chen. "Integrating mesh and meshfree methods for physics-based fracture and debris cloud simulation." In M. Botsch, B. Chen, M. Pauly, and M. Zwicker, editors, *Symp Point-Based Graph.* Eurograph Assoc, 2006.