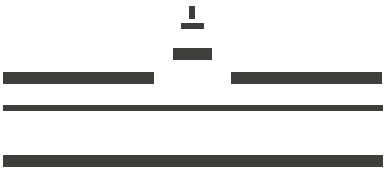


Westfälische
Wilhelms-Universität
Münster

Efficient Evolution Algorithms for Embedded Interfaces

From Inverse Parameter Estimation to a Level Set Method for Ductile Fracture

Jan Hegemann

- 2013 -

wissen.leben
WWU Münster

Westfälische
Wilhelms-Universität
Münster

Efficient Evolution Algorithms for Embedded Interfaces

From Inverse Parameter Estimation to a Level Set Method for Ductile Fracture

Fach: Mathematik

Inaugural-Dissertation zur Erlangung des
Doktorgrades der Naturwissenschaften
- Dr. rer. nat. -
im Fachbereich Mathematik und Informatik
der Mathematisch-Naturwissenschaftlichen Fakultät
der Westfälischen Wilhelms-Universität Münster

vorgelegt von
Jan Hegemann
- 2013 -

wissen.leben
WWU Münster

Dekan:	Prof. Dr. Martin Stein
Erster Gutachter:	Prof. Dr. Martin Burger (Westfälische Wilhelms-Universität Münster)
Zweiter Gutachter:	Prof. Dr. Joseph Teran (University of California, Los Angeles)
Tag der mündlichen Prüfung:	
Tag der Promotion:	

ABSTRACT

This thesis presents efficient and easy to implement algorithms for the simulation of embedded interface evolutions in applications of physically based modeling. It is organized in two parts.

In the first part, we provide the mathematical and numerical background necessary for the development of our methods. We outline a state-of-the-art cut-cell algorithm that allows us to efficiently and conveniently embed level set surfaces into a regular mesh, eliminating the need for boundary aligned elements. An implicit discretization of dynamic non-linear elasticity is derived, which combines the finite element method with the aforementioned embedded mesh approach.

The second part is concerned with the development of specialized algorithms to specific problems involving embedded surfaces. First, we introduce a general and efficient method to recover piecewise constant coefficients occurring in elliptic partial differential equations as well as the interface where these coefficients have jump discontinuities. For this inverse problem, we use an output least squares approach with level set and augmented Lagrangian methods. Our formulation incorporates the inherent nature of the piecewise constant coefficients, which eliminates the need for a complicated non-linear solve at every iteration. Instead, we obtain an explicit update formula and therefore vastly speed up computation. We employ our approach to the example problems of Poisson's equation and linear elasticity, and simultaneously recover both coefficients and interface for these problems.

In another application of the background developed in the first part, we utilize the shape derivative of the classical Griffith's energy of fracture mechanics in a level set evolution for the simulation of dynamic ductile fracture. The level set is defined in the undeformed configuration of the object, and its evolution is designed to represent a transition from undamaged to failed material. No re-meshing is needed since the resulting topological changes are handled naturally by the level set method. We provide a new mechanism for the generation of fragments of material during the progression of the level set in the Griffith's energy minimization. Collisions between different material pieces are resolved with impulses derived from the material point method over a background Eulerian grid. This provides a stable means for colliding with embedded interfaces. We demonstrate the efficacy of our algorithm by presenting compelling results of many realistic-looking fracture simulations, which may be used in computer graphics and movies.

Key words: evolving interfaces, efficient algorithms, level set method, finite element method, physically based modeling, inverse constitutive modeling, inverse problem, elliptic problem, parameter estimation, Poisson equation, linear elasticity, augmented Lagrangian method, ductile fracture, collisions, corotational elasticity, plasticity

ACKNOWLEDGEMENTS

I thank

Martin Burger for being my PhD adviser and for giving me the opportunities I had: sending me to UCLA for my Diploma thesis, which ultimately led to the research presented in this dissertation, for allowing me to do most of this research at UCLA and yet graduating from WWU Münster, and for making it possible for his students - me amongst them - to pursue their own interests and personal strengths;

Joey Teran for being my adviser at UCLA, for introducing me to the exciting field of physics-based modeling and graphics, for asking me to come back after my Diploma thesis and continue my work as a PhD student, for always pushing for progress, and for providing research opportunities unique to UCLA;

Chenfanfu Jiang for being a hard-working and productive co-author, and for finishing a very cool project together;
Jeff Hellrung for being my go-to C++ guy and for his patience whatever the question;
Craig Schroeder for productive discussions about elasticity and implementations;
Frank Wübbeling for his help with computational resources to finish my last project while in Münster;
Paul Bunn, Rachel Hegemann, Pia Heins, Jeff Hellrung, and Christoph Brune for carefully reading the manuscript and their help in editing this dissertation;

the UCLA Mathematics Department for welcoming me as a long-term visiting researcher; everyone in the Institute for Numerics and Applied Mathematics in Münster for making the department such a friendly as well as productive work environment;

all friends near and far for being there for me during the course of my studies.

Finally, I feel deep gratitude towards

my parents, Ingrid and Dieter, and brother Arne for allowing me to become who I am today;

my smart and beautiful wife Rachel Hegemann: without your love and support during tough times I would have never made it this far.

Contents

List of Figures

List of Algorithms

List of Tables

1 Introduction

1.1 Motivation

The process of earning a PhD is often considered a journey, a quest for knowledge as well as for personal growth. My journey was even a physical one, and it started before I even thought about towards a doctorate. In March 2008 I was lucky enough to depart for a year to study abroad at the University of California, Los Angeles (UCLA). Upon arrival, I was offered the great opportunity to talk to many of the researchers in the Department of Mathematics there and given the choice to freely decide which project interested me the most. Certainly, all of them had their own appeal, from swarm simulations and robots to crime modeling and many more. One conversation stood out: Professor Joseph Teran caught my attention by offering me the prospect to work on fracture mechanics and how it could lead to applications in graphics, namely in special effects and animations for movies. For a cinephile like myself, that temptation was simply too great to ignore! To put it candidly: I was hooked.

However, it would be a long way to realize this goal of fracture for graphics. The first project instead started with the simulation of two dimensional quasistatic, brittle material, with application in engineering. This project led to not only my Diploma thesis ([54]) but also to my first publication ([119]). One noteworthy aspect and the reason I recall it here is the method of *crack propagation*. As is commonplace in the literature of this specific field, we used a fracture criterion that relies on the computation of a so called propagation angle, which, in turn, is determined from the current deformation of the material. However, while this angle changes according to the current state and the current stress, the length of each increment is fixed and determined a priori. In other words, the propagation speed is always the same. Furthermore, this approach does not allow the crack to split or merge. This inflexibility was something that I wanted to overcome from the beginning - but I did not yet have the appropriate tools to do so. During this first project, and before having accomplished hardly anything yet, Joseph

Teran asked me to continue my work as a PhD student, and I was thankful that Martin Burger agreed to make it possible despite some logistical impediments.

After graduating with my Diploma in August 2009 in Münster, I went back to Los Angeles the following month and was excited to continue my research. A new project was quickly found, and it combined a multitude of very interesting mathematical aspects: it dealt with an inverse problem, for which my home Institute for Computational and Applied Mathematics at the Westfälische Wilhelms-Universität Münster is well positioned; it investigated linear elasticity, with which I was already quite accustomed from my Diploma thesis; and it featured level set methods, which had interested me since I had first learned about them in a seminar a few years prior. This project was successful (not without its own obligatory ups and downs, of course); it resulted in my first first-author publication and is part of this dissertation in Chapter 3.

During this project I learned a lot about evolving level sets and how to use them in combination with shape optimization techniques. This finally gave me the tools I had been looking for: it opened a way to avoid the restrictions imposed by many other crack propagation methods by allowing completely arbitrary fracture evolutions, with a variable propagation speed and without any requirements on a priori crack initialization. Consequently, we once again approached the problem of fracture mechanics, this time under the premise of developing an efficient and versatile propagation method for highly dynamic simulations, with inertia effects and in 3D. After countless hours of work, and certainly some more complications, I was finally where I had wanted to arrive: movie-grade fracture! The results of this work can be found in Chapter 4.

With the conclusion of this project, my pursuit of a doctorate degree comes to an end as well. It was a hard one, a joyful one, one that was frustrating at times, but also exciting as well as certainly humbling - all aspects inherent to mathematics itself in one form or another, however as mentioned before, also to the PhD process in general.

Mathematically speaking, the journey revolves around evolving interfaces. This is a phenomenon that occurs all around us, from the macroscopic growth of living organism, the branching of a tree, to the cells that go through mitosis at the micro scale, in a rapidly moving and splashing body of water or the growth of crystals. While we may marvel at these wonders of nature, the mathematical description of their surfaces can prove to be non-trivial due to changes of their form, including branching, opening and closing of holes, etc. As one uses the derivative of a function to analyse its rate of change, we will use a similar approach to understand the change in the shape of a surface. This allows the development of optimization algorithms whose objective is to minimize a suitable energy as a function of an evolving shape.

The next stop on this mathematical venture are so called *inverse problems*. While the forward problem is the task of finding a solution to a problem for which one is given all the model parameters (the equation, initial and boundary conditions, etc.), the inverse problem is concerned with the recovery of parts of the equation from a given observation of the solution. An intuitive example would be an object under some lighting conditions: The forward problem solves for the shape of the shadow from the geometry of the object. Conversely, the inverse problem tries to infer the geometry of the object from a given shadow. As is obvious from the example, the inverse problem is often very difficult to solve and might not have a unique solution. Additional information, like the shadows of the same object under different angles of light incidence, can improve the conditions for finding a solution. The difficulty that frequently arises when dealing with inverse problems is called the *ill-posedness* of the problem. It was first categorized by Jacques Hadamard in 1902. According to his work, a problem is *well-posed* if it meets the following three conditions:

1. A solution exists.
2. The solution is unique.
3. The solution depends continuously on the initial conditions, in other words, the solution's behavior hardly changes when there is a slight change in the initial conditions.

Problems not fulfilling these requirements are called *ill-posed*. Inverse problems have a well-established place in modern science and technology. They are crucial to applications like radar, sonar, medical imaging (e.g. computer tomography), image processing techniques (e.g. denoising and debluring), analysis of properties of astronomical objects based on the emitted or reflected light, unmixing problems (e.g. spectrography), and many more.

The final stop in this journey is in creating *physically based simulations for computer graphics*. Once again, many examples can be found in well-known applications. Almost every modern professional movie with special effects contains computer generated images (CGI); nowadays animated movies are often fully rendered by computers; and computer games using photo-realistic graphics combined with physically correct behavior have arrived in the mainstream. And in the future, computer graphics can even help to advance medicine via visualization of internal organs or by providing customized virtual surgery environments as a method of preparation before major operation, individualized for every patient.

During the creation of an animated movie, instead of drawing everything by hand, artists often utilize computer simulations. They model the objects and characters, from the skeleton over body mass to hair and clothing, directly in the computer. Numerical

mathematics are then used to realistically simulate their behavior when set in motion. In the example of a character, the motion of only the skeleton and muscles might be specified by the artist, and then the body mass, hair and clothing all move as a result. This computer assistance allows the animators to spend more time on creative aspects of the story, while the simulation handles providing more realism. However, it also takes full control out of their hands and fully realistic behavior might not be desirable at all times. Instead, it only needs to look realistic for the viewer and artistic considerations may sometimes be more important than physical correctness. This motivates the incorporation of controlling features into the numerical methods. Furthermore, efficiency is vital for these applications. While it is true that computers steadily become faster, the computational budget for a set project typically remains the same. Thus, an efficient method can improve the appearance of the final result since it allows the simulation of a more complex object within the same time frame. For a real-time application like video games or virtual surgery simulators efficiency is even key to their feasibility.

1.2 Organization of this Work and Contributions

This thesis is organized as follows:

In Chapter 2, we present the theory and background of the mathematical tools necessary to develop our methods. It includes results from shape optimization as well as many different numerical concepts and methods, such as the level set method, the development of a level set based embedded meshing algorithm and a derivation of a finite element discretization of dynamic elasticity. This first part lays the foundation to understand and follow the methods we develop in the second part of this thesis. Implementation details are given where they apply to both our algorithms.

The second part of this thesis, consisting of Chapters 3 and 4, can be considered its main part. It is concerned with applications based on the fundamentals outlined earlier. For these applications we elaborate and expand on the basis given before and investigate additional concepts as they are needed specifically for each problem. Details to additional, advanced implementation aspects are also presented.

The first of the two applications is the inverse problem of parameter and interface estimation of elliptic partial differential equations (PDEs). In our case, we concern ourselves with the two well-known problems of Poisson's equation and linear elasticity. We assume to have solutions to each of these PDEs and try to recover the values of the coefficients that constitute the equations. Since we allow these coefficients to have jump discontinuities, we also recover the interface where these jumps occur. Our contribution

is an efficient and easy to implement algorithm. We accomplish this by incorporating the piecewise constant nature of the coefficients directly into the method. Numerical results for both equations and different 2D geometric setups are given, and several noise levels are investigated.

The second application is the aforementioned dynamic fracture for graphics applications. We generalize an energy-driven level set based evolution to dynamic non-linear elasticity, which yields realistic fracture simulations. We provide a new mechanism for the generation of fragments of material during the progression of the level set in the energy minimization. Collisions between different material pieces are resolved with impulses derived from the material point method over a background Eulerian grid. With this extension of the material point method, we present a solution to the problem of collisions between embedded surfaces. Our resulting algorithm is again easy to implement and efficient to execute. It also allows us to control the fracture propagation speed and if desired even guide its path. Results for both 2D and 3D are presented, ranging from stretching of block to hitting complex geometries with multiple projectiles. Dynamics are fully simulated and yield compelling effects, including secondary fracture resulting from collisions of fragment pieces with the material.

Since this work is heavily based on numerical methods and algorithmic considerations, we here list some facts about the programming work done for this dissertation. The results of this aspect are shown in the respective subsections of Chapters 3 and 4.

- programming in Microsoft Windows as well as Linux environments
- over 17,000 lines of C++ code developed
- substantial contributions to many different aspects of the scientific computing library called Sake, local to the Teran work group at UCLA
- (almost) everything beyond basic data structures was newly written: from discretization over solver to level set advection and reinitialization
- extensive use of sparse and efficient data structures, iterators etc. (for matrices, cutting algorithms, collision handling and more)
- utilization different libraries: STL, Boost[1] , PETSc[2]
- heavy use of multi-threading with OpenMP[3]
- handling output to .vtk files, including writing customized interfaces to this format, and visualization with Paraview[4]

[1]Boost C++ Libraries: `http://www.boost.org/`
[2]PETSc (Portable, Extensible Toolkit for Scientific Computation): `http://www.mcs.anl.gov/petsc/developers/projects.html`
[3]OpenMP (API for multi-platform shared-memory parallel programming in C/C++ and Fortran): `http://www.openmp.org/`
[4]Paraview (open source scientific visualization): `http://www.paraview.org/`

- rendering with POV-ray[5]

[5]POV-ray (the Persistence of Vision Raytracer): `http://www.povray.org/`

2 Background

In this chapter we will present all the background required to understand the methods we will develop in the later chapters. This background consists of both theoretical results as well as numerical concepts and algorithms. The goal is to lay the foundation consisting of the components that are common to each of our methods.

First, we introduce the definition and some essential results of *shape derivatives*. These will be integral to both our methods and build the cornerstone of our evolving interfaces. Those interfaces will be represented and evolved using the *level set method* as described in Section 2.2. The implicit description of the interface inherent to this method allows to handle any occurring topological changes naturally. One minor drawback is that the level set function needs to be reinitialized as a signed distance function, a process we describe in Section 2.2.1.

We use the information provided by the interface representation to *embed* the surface into a regular triangle or tetrahedron mesh (in 2D and 3D respectively). This process, which we detail in Section 2.3, builds the ground infrastructure of our discretization. It allows for accurate computation of mesh-based forces while eliminating the necessity for remeshing every time the interface moves. Instead, integrals are evaluated over the material volume of partially filled elements.

In Section 2.4 we step through all aspects of the *elasticity discretization* according to the finite element method (FEM). We use a fully implicit backward Euler scheme to allow for larger time steps. The resulting non-linear system is solved via Newton's method. Lastly, we show how to *solve* the occurring symmetric but indefinite linear systems via the minimal residual method (MINRES) in Section 2.5.

2.1 Shape Derivative

Since our methods investigate and use the evolution of the boundary of a domain (or subset thereof), we first need to understand how a functional, which is dependent on

the domain, behaves under these changes. Therefore, we first need to define the *shape derivative*.

Definition 2.1 (Shape Derivative)**.** *Let $\Omega \subset \mathbb{R}^d$ be open and bounded. The shape derivative of a functional $J(\Omega)$ at Ω is defined as the Frechét derivative in $W^{1,\infty}(\mathbb{R}^d;\mathbb{R}^d)$ at 0 of the application $\theta \mapsto J((I+\theta)(\Omega)) = \{x+\theta(x)|x \in \Omega\}$, i.e.*

$$J((I+\theta)(\Omega)) = J(\Omega) + J^{'}(\Omega)[\theta] + o(\theta),$$

where $J^{'}(\Omega)$ is a continuous linear form on $W^{1,\infty}(\mathbb{R}^d,\mathbb{R}^d)$ and $\lim_{\theta\to 0} \frac{|o(\theta)|}{||\theta||} = 0$.

We now derive two essential statements about shape derivatives (cf. [2, 131]) and use them in the subsequent investigations. The first one provides the means to compute the derivative of a function integral with respect to the region of integration.

Lemma 2.2. *Let $\Omega \subset \mathbb{R}^d$ be a smooth open set and and $f \in W^{1,1}(\mathbb{R}^d)$. Then*

$$J(\Omega) := \int_\Omega f(x)\ dx$$

is differentiable at Ω and

$$\begin{aligned} J'(\Omega)[\theta] &= \int_\Omega \nabla \cdot (\theta(x) f(x))\ dx \\ &= \int_{\partial\Omega} \theta(x) \cdot n(x) f(x)\ ds(x) \end{aligned}$$

for any $\theta \in W^{1,\infty}(\mathbb{R}^d;\mathbb{R}^d)$.

Proof. Let $\Omega^{'} = (I+\epsilon\theta)(\Omega)$ be a small perturbation of Ω in the direction of θ, and let γ: $\mathbb{R}^d \to \mathbb{R}^d$, $\Omega \mapsto \Omega^{'}$ be the mapping between the two sets, i.e. $\gamma(\boldsymbol{X}) = (I+\epsilon\theta)(\boldsymbol{X})$. Then

$$\begin{aligned} J((I+\epsilon\theta)(\Omega)) &= \int_{\Omega'} f(y)\ \mathrm{d}y \\ &= \int_\Omega f(\gamma(x))\ \det(\mathrm{D}\gamma)(x)\ \mathrm{d}x, \end{aligned}$$

where $\mathrm{D}\gamma$ denotes the Jacobian of γ. The directional derivative of J is then given by

$$\begin{aligned} J'(\Omega)[\theta] &= \frac{\partial}{\partial\epsilon}|_{\epsilon=0} J((I+\epsilon\theta)(\Omega)) \\ &= \frac{\partial}{\partial\epsilon}|_{\epsilon=0} \int_\Omega f(\gamma(x))\ \det(\mathrm{D}\gamma)(x)\ \mathrm{d}x \\ &= \int_\Omega \nabla f(x)\cdot\theta(x) + f(x)\ \nabla\cdot\theta\ \mathrm{d}x \\ &= \int_\Omega \nabla\cdot(f\theta)\ \mathrm{d}x \end{aligned}$$

where we used $\det(\mathrm{D}\gamma)|_{\epsilon=0} = \det(\mathrm{D}(I+\epsilon\theta))|_{\epsilon=0} = 1$ as well as

$$\begin{aligned} \frac{\partial}{\partial\epsilon}|_{\epsilon=0}\ \det(\mathrm{D}\gamma) &= \frac{\partial}{\partial\epsilon}|_{\epsilon=0}\ \det(\mathrm{D}(I+\epsilon\theta)) \\ &= \frac{\partial}{\partial\epsilon}|_{\epsilon=0}(1+\epsilon\nabla\cdot\theta+O(\epsilon^2)) \\ &= \nabla\cdot\theta \end{aligned}$$

and $\nabla f\cdot\theta + f\nabla\cdot\theta = \nabla\cdot(f\theta)$ for any differentiable scalar valued function f and vector field θ. The divergence theorem completes the proof. □

Remark: The above lemma can also be interpreted as a special case of the Reynold's transport theorem

$$\frac{d}{d\tau}\int_{\Omega(\tau)} f dV = \int_{\Omega(\tau)} \frac{\partial f}{\partial\tau} dV + \int_{\partial\Omega(\tau)} (\boldsymbol{v}^b\cdot\boldsymbol{n}) f dA,$$

for an f that is independent of the change of the domain Ω, i.e. independent of τ, and if one interprets θ from above as the change to the domain, i.e. its velocity $\boldsymbol{v}^b$.

The next lemma contains the well-known result that the derivative of the length of a curve is given by its curvature - which is precisely the part we will be using in our subsequent investigations.

Lemma 2.3. *Let $\Omega\subset\mathbb{R}^d$ be a smooth open set and $f\in W^{2,1}(\mathbb{R}^d)$. Then*

$$J(\Omega) := \int_{\partial\Omega} f(x)\ ds$$

is differentiable at Ω and

$$J'(\Omega)[\theta] = \int_{\partial\Omega} \theta(x)\cdot n(x)\left(\frac{\partial f}{\partial n}(x) + f(x)\kappa(x)\right)\ ds$$

for any $\theta \in W^{1,\infty}(\mathbb{R}^d;\mathbb{R}^d)$*, where* $\kappa = \nabla \cdot n$ *is the mean curvature of* $\partial\Omega$*.*

Proof. We refer the interested reader to [131], page 80. □

2.2 Level Set Evolution

We will only give a brief description of the level set method and refer to [106] for a more detailed survey. The level set method was first developed by Osher and Sethian in [107] and has since become a popular and powerful tool in many applications, ranging from image segmentation (e.g. [147]), to dynamic fluid, smoke and fire simulations (see [85] and references therein).

The idea of the level set method is as simple as it is effective: instead of an explicit representation of a surface, we model it as the zero isocontour of a (continuous) function ϕ, i.e. as the set $\Gamma := \{\boldsymbol{X} \in \mathbb{R}^d \mid \phi(\boldsymbol{X}) = 0\}$. One intuitive example is a circle with radius r around the origin, described by the equation

$$X_1^2 + X_2^2 - r^2 = 0 \tag{2.1}$$

This formulation has many powerful advantages. First, the sign of ϕ at any given point easily indicates which side of the interface this point is on. Further, the implicit representation allows for natural handling of changes in the topology of the interface, such as splitting and merging, because these can be executed by changing the definition of the function values. This versatility makes it perfectly suited for problems like inverse shape reconstruction (see Chapter 3) and fracture propagation (see Chapter 4), where topological changes occur frequently.

The level set function ϕ can be evolved throughout an artificial time $s \in \mathbb{R}^+$ by computing the total derivative

$$\frac{\mathrm{d}\phi}{\mathrm{d}s}(\boldsymbol{X}(s), s) = \frac{\partial\phi}{\partial s} + \nabla\phi \cdot \boldsymbol{v}. \tag{2.2}$$

There are many choices for the velocity field $\boldsymbol{v}$; classical ones are mean curvature or a flow field. It can also be chosen - and we will elaborate on this later - as a descent direction of an energy functional. In that case, the interface (hopefully) moves towards a stationary point. The velocity in the gradient descent direction can be found using the lemmata from Section 2.1 (using the same notation) by setting

$$\theta(x) = -f(x)n(x). \tag{2.3}$$

Its normal component $V := \theta \cdot n (= f)$ can then be used as the advection velocity in the

Hamilton-Jacobi equation

$$\frac{\partial \phi}{\partial s} - |\nabla \phi| V = 0, \tag{2.4}$$

where we also used $n = \frac{\nabla \phi}{|\nabla \phi|}$. Equation (2.4) is called the *level set equation*.

We further replace $|\nabla \phi|$ with an approximation of the Dirac delta-function (see [147]):

$$\frac{\partial \phi}{\partial s} - \ \delta_\epsilon(\phi) V = 0. \tag{2.5}$$

One possible choice is the following C^∞ representation (see [147]):

$$\delta_\epsilon(x) = \frac{1}{\pi} \frac{\epsilon}{\epsilon^2 + x^2}. \tag{2.6}$$

The shape optimization can then be implemented with a forward-Euler scheme, yielding

$$\phi^{k+1}(\boldsymbol{X}) = \phi^k(\boldsymbol{X}) + \Delta s \, \delta_\epsilon(\phi^k(\boldsymbol{X})) \, V(\boldsymbol{X}). \tag{2.7}$$

which evolves ϕ with some time step Δs.

2.2.1 Reinitialization

The most common choice for the level set function ϕ is that of a *signed distance function*, i.e.

$$\phi(\boldsymbol{X}) = \begin{cases} \min\limits_{\boldsymbol{X}_0 \in \Gamma} (|\boldsymbol{X} - \boldsymbol{X}_0|) & \text{if } \boldsymbol{X} \in \Omega^+ \\ - \min\limits_{\boldsymbol{X}_0 \in \Gamma} (|\boldsymbol{X} - \boldsymbol{X}_0|) & \text{if } \boldsymbol{X} \in \Omega^- \end{cases}, \tag{2.8}$$

where Ω^+ denoted the domain enclosed by the interface Γ and Ω^- the one outside of it, i.e. $\Omega^- = \Omega \setminus (\Omega^+ \cup \Gamma)$. Reformulating the example of the circle given in (2.1) in this fashion yields

$$\sqrt{X_1^2 + X_2^2} - r = 0. \tag{2.9}$$

One of the properties of a signed distance function is that $|\nabla \phi| = 1$, which simplifies the level set equation (2.4). However, it is well known (e.g. [106, 113]) that the solution to the evolution given by (2.4) is generally not a signed distance function any more, even if the initial ϕ is. Instead, the solution can become too flat or too steep around the interface. The process of re-establishing the signed distance property is called *reinitialization*. This procedure of replacing the function ϕ is theoretically justified in [31] where it is shown that the interface at time s, $\Gamma(s) := \{\boldsymbol{X} \in \mathbb{R}^d \mid \phi(\boldsymbol{X}, s) = 0\}$, given by the solution

of equation (2.4) depends only on the initial interface $\Gamma(0) := \{\boldsymbol{X} \in \mathbb{R}^d \mid \phi(\boldsymbol{X}, 0) = 0\}$ but not on the particular choice of the initial function $\phi(\cdot, s)$. While in theory reinitialization is necessary after every change to ϕ, numerically it might be beneficial not doing so - an example and more details about this can be found in Chapter 3.

A straight-forward way to reinitialize ϕ as signed distance is to approximate the location of the interface (e.g. by interpolation), then recompute the distance between all grid nodes and this surface ([91]). However, doing this for the entire domain Ω is computationally expensive and might lead to a non-smooth ϕ. A more elegant way is to solve the eikonal equation

$$|\nabla\phi(\boldsymbol{X})| = f(\boldsymbol{X}) \quad \boldsymbol{X} \in \Omega \subset \mathbb{R}^d \tag{2.10}$$

$$\phi(\boldsymbol{X}) = 0 \qquad \boldsymbol{X} \in \Gamma \subset \Omega \tag{2.11}$$

for the special case of

$$f(\boldsymbol{X}) \equiv 1. \tag{2.12}$$

This can be done with various discretization schemes (e.g. [113, 137]). A computationally efficient method is the fast sweeping method described in [149], which we will summarize here for completeness.

Assuming we have ϕ defined on an equidistant grid with spacing h and grid indices (i, j, k), the basic idea is to update the values of ϕ as the minimum of its current value and the solution of a Godunov-type upwind scheme

$$[(\phi_{ijk} - a_1)^+]^2 + [(\phi_{ijk} - a_2)^+]^2 + [(\phi_{ijk} - a_3)^+]^2 = h^2 \tag{2.13}$$

where

$$a_1 := \min(\phi_{(i-1)jk}, \phi_{(i+1)jk}) \tag{2.14}$$

$$a_2 := \min(\phi_{i(j-1)k}, \phi_{i(j+1)k}) \tag{2.15}$$

$$a_3 := \min(\phi_{ij(k-1)}, \phi_{ij(k+1)}) \tag{2.16}$$

$$x^+ := \max(0, x). \tag{2.17}$$

For boundary nodes one-sided differences are used instead of equations (2.14) - (2.16).

The solution is propagated through Gauss-Seidel iterations in multiple "sweeping steps". The boundary condition is enforced by initializing and fixating the values around the interface. If the level set evolution is sufficiently well behaved, this can be done by using the values resulting from the solution to (2.4) for the nodes directly adjacent to

the interface. Alternatively, the distance to the interface can be recomputed for these nodes.

For all nodes not adjacent to the interface, we first save their sign and then initialize their value to a large value c_{large} with the property

$$M_\phi \leq c_{\text{large}} \leq \infty, \tag{2.18}$$

where M_ϕ is the largest possible value for ϕ in the entire domain Ω. Since any such value is sufficient, a numerical representation of positive infinity can be used in practice for c_{large}.

All grid nodes initialized with c_{large} will be updated by finding the solution to (2.13). First, we order the a_ℓ defined by (2.14) - (2.16), such that $a_1 \leq a_2 \leq a_3 < a_4 := \infty$. Then, there is a p such that $\bar{x}$ is the unique solution of

$$(x - a_1)^2 + \cdots + (x - a_p)^2 = h^2 \tag{2.19}$$

that satisfies

$$a_p < \bar{x} \leq a_{p+1}. \tag{2.20}$$

We find $\bar{x}$ by the following iterative process: For $p = 1$, define $\tilde{x}$ as the unique solution of

$$(x - a_1)^2 = h^2 \tag{2.21}$$

under the requirement

$$\tilde{x} > a_1, \tag{2.22}$$

which obviously is

$$\tilde{x} = a_1 + h. \tag{2.23}$$

If $\tilde{x} \leq a_2$, then $\bar{x} = \tilde{x}$. If not, find the unique solution $\tilde{x}$ of

$$(x - a_1)^2 + (x - a_2)^2 = h^2 \tag{2.24}$$

with

$$\tilde{x} > a_2, \tag{2.25}$$

computed as

$$\tilde{x} = \frac{a_1 + a_2 + \sqrt{2h^2 - (a_1 - a_2)^2}}{2}. \tag{2.26}$$

In two dimensions, we are guaranteed to have found our solution at this point. In 3D we continue in the obvious fashion: If $\tilde{x} \leq a_3$, then $\bar{x} = \tilde{x}$, else find $\tilde{x}$ as the unique solution of

$$(x - a_1)^2 + (x - a_2)^2 + (x - a_3)^2 = h^2 \tag{2.27}$$

that satisfies

$$\tilde{x} > a_3, \tag{2.28}$$

given by

$$\tilde{x} = \frac{\sqrt{(-2a_1 - 2a_2 - 2a_3)^2 - 12(a_1^2 + a_2^2 + a_3^2 - h^2)} + 2a_1 + 2a_2 + 2a_3}{6}, \tag{2.29}$$

which concludes this part of the algorithm. This scheme can be generalized to higher dimensions if desired.

The correct distance is propagated from the interface outwards by employing multiple sweeping steps in all possible combination of directions:

$$(1) \quad i = 1 : N_i, \quad j = 1 : N_j \tag{2.30}$$
$$(2) \quad i = 1 : N_i, \quad j = N_j : 1 \tag{2.31}$$
$$(3) \quad i = N_i : 1, \quad j = 1 : N_j \tag{2.32}$$
$$(4) \quad i = N_i : 1, \quad j = N_j : 1 \tag{2.33}$$

in two dimensions, and in 3D all 2^3 directions

$$(1) \quad i = 1 : N_i, \quad j = 1 : N_j, \quad k = 1 : N_k \tag{2.34}$$
$$(2) \quad i = 1 : N_i, \quad j = 1 : N_j, \quad k = N_k : 1 \tag{2.35}$$
$$(3) \quad i = 1 : N_i, \quad j = N_j : 1, \quad k = 1 : N_k \tag{2.36}$$
$$(4) \quad i = 1 : N_i, \quad j = N_j : 1, \quad k = N_k : 1 \tag{2.37}$$
$$(5) \quad i = N_i : 1, \quad j = 1 : N_j, \quad k = 1 : N_k \tag{2.38}$$
$$(6) \quad i = N_i : 1, \quad j = 1 : N_j, \quad k = N_k : 1 \tag{2.39}$$
$$(7) \quad i = N_i : 1, \quad j = N_j : 1, \quad k = 1 : N_k \tag{2.40}$$
$$(8) \quad i = N_i : 1, \quad j = N_j : 1, \quad j = N_k : 1, \tag{2.41}$$

where N_i, N_j, N_k are the number of grid nodes in $x = X_1, y = X_2$, and $z = X_3$ direction respectively.

Lastly, we re-establish the correct signs for all nodes. Since these were saved in the initialization phase, a simple loop is sufficient. The entire method then reads as shown in Algorithm 2.1.

The main advantage of this algorithm is its optimal complexity of $\mathcal{O}(N)$ for N grid nodes and that it converges after 2^d sweeps if executed in the order as outlined above. For an analysis of the convergence behaviour as well as error estimates, we again refer to the original work of [149].

2.3 Level Set Based Embedded Meshing

We will now explain how we incorporate the implicit representation of the level set method into our discretization. Similar to [82], we use a signed distance function ϕ in material coordinates $\boldsymbol{X}$ to create an embedded Lagrangian mesh. For any given domain $\Omega_0 \subset \mathbb{R}^d$ and its interface $\Gamma = \partial\Omega_0$, we create a larger domain Ω_* that encloses Ω_0 but is of regular shape and easy to tessellate. A convenient choice for Ω_* is the bounding box of Ω_0. We then start with a simple regular lattice $\mathcal{G}_h$ of Ω_* with spacing $h := \Delta x = \Delta y \ (= \Delta z)$. We define the values of the level set function ϕ at the nodes of $\mathcal{G}_h$. The computational mesh $\mathcal{M}_h$ finally consists of all elements in $\mathcal{G}_h$ with at least one node $\boldsymbol{X}_i$ having $\phi(\boldsymbol{X}_i) < 0$. Some of the elements will have negative ϕ values on all nodes, we call these *inside elements*. Integration on these elements will be independent of the embedding. Elements that contain both nodes with positive and negative level set values will be denoted as *boundary elements*. These elements are partially filled with material and the integration will reflect that (see below). We refer to any node incident on a boundary element that has a positive ϕ value as a *virtual node* since it is outside but still participates in the discretization by virtue of the embedding. In other words, it is located outside of the domain, however, some portion of the material is associated with this node. See Figure 2.1 for a 2D example.

In practice, we only use simplices, i.e. triangles in 2D and tetrahedra in 3D, which correspond to elements with linear basis functions in the FEM framework (see Section 2.4 below), and we will focus our language on this case. Let us suppose that $\Omega_0 \subset \mathbb{R}^d$ is tesselated by simplices $\mathcal{M}_h = \{S^1, \ldots, S^N\}$ with grid vertices $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^n \in \mathbb{R}^d$ and with S_k^α denoting the kth grid vertex of simplex S^α. We create a linear approximation of the sub-element location of the zero isocontour to define the boundary of the material region. That is, we introduce either a triangle or (convex) quadrilateral (simple line segments in 2D) on each boundary element depending on the number of edge crossings, as we will explain now.

Algorithm 2.1 Fast-sweeping.

```
// Input: φ, h
// Initialization
for all i = 1 to N_i do
  for all j = 1 to N_j do
    for all k = 1 to N_k do
      signs(i,j,k) ← sign(φ_ijk)
      if φ_ijk is incident to interface then
        fixed_nodes(i,j,k) ← true
        φ_ijk ← |φ_ijk|
      else
        φ_ijk ← c_large
      end if
    end for
  end for
end for

// Sweeping steps
for all sweeps = 1 to 2^d do
  // loop directions as outlined in (2.30) - (2.33) or (2.34) - (2.41)
  for all i do
    for all j do
      for all k do
        if not fixed_nodes(i,j,k) == true then
          φ̄_ijk = upwind_solution(φ_ijk)
          φ_ijk ← min(φ̄_ijk, φ_ijk)
        end if
      end for
    end for
  end for
end for

// Re-establishing the signs
for all i = 1 to N_i do
  for all j = 1 to N_j do
    for all k = 1 to N_k do
      φ(i,j,k) ← signs(i,j,k) · φ_ijk
    end for
  end for
end for
```

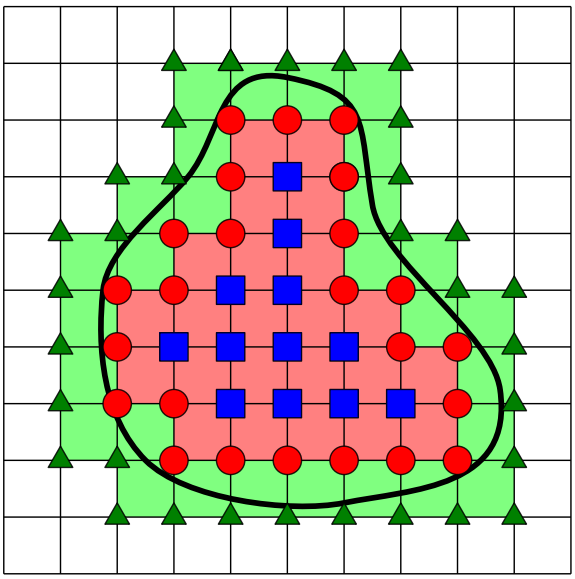


Figure 2.1: Our level set based embedding in a regular grid in 2D. Boundary cells are shown in green, virtual nodes are depicted with green triangles, nodes that have a discrete stencil independent of embedding are in blue.

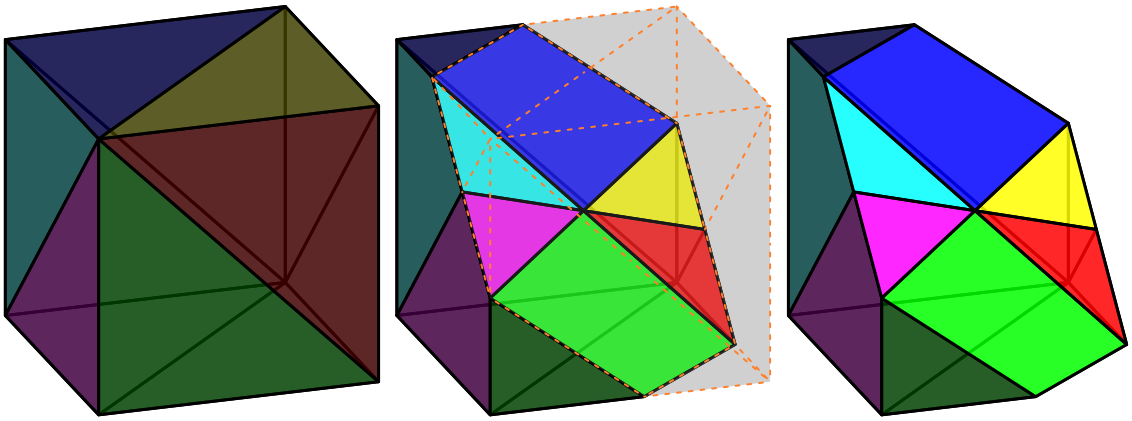
For a boundary element $S^\alpha \in \mathcal{M}_h$, we check each edge, which we identify by the pair of incident indices, $(i = S^\alpha_k, j = S^\alpha_\ell)$. If $\phi(\boldsymbol{X}_i)\phi(\boldsymbol{X}_j) < 0$, we locate the interface Γ on this edge via the intersection point. Its barycentric coordinates $\boldsymbol{w} = (w_k)_k \in \mathbb{R}^{d+1}$ are defined by

$$w_k = \frac{\phi(\boldsymbol{X}_j)}{\phi(\boldsymbol{X}_i) - \phi(\boldsymbol{X}_j)} \tag{2.42}$$

$$w_\ell = \frac{\phi(\boldsymbol{X}_i)}{\phi(\boldsymbol{X}_i) - \phi(\boldsymbol{X}_j)}; \tag{2.43}$$

all other components of $\boldsymbol{w}$ are equal to zero (cf. [92]). Note that since $\sum_{k=1}^{d+1} w_k = 1$, it suffices to store only the d independent entries in order to save memory. Further, absolute values are not needed in the computation of (2.42) and (2.43) as $\phi(\boldsymbol{X}_i)$ and $\phi(\boldsymbol{X}_j)$ are of opposite sign, thus the above always results in the desired positive weights.

Now, in 3D there are two distinct possibilities in any boundary element (in 2D only one trivial case occurs): (i) three vertices are of the same sign and the fourth is of the opposite or (ii) two each share the same sign. The first case leads to three edge crossings and therefore a triangle as material surface in this element, the second results in four intersections and thus a quadrilateral insertion. Note that the vertices of the embedded surface are not degrees of freedom in our discretization. Only the element nodes themself give rise to degrees of freedom, and the intersection points are fully dependent on those vertices via their barycentric coordinates. Figure 2.2 shows a 3D illustration of this division process and the resulting material portion.



(a) Cube divided into tetrahedra. (b) Tetrahedra cut by a level set. (c) Material portion of cut tetrahedra.

Figure 2.2: Illustration of our tetrahedralization and embedding on a 3D uniform grid. Each cell is subdivided into 6 tetrahedra. The linear interpolation of the interface results in the addition of either a triangular or quadrilateral embedded surface in every cut element.

2.3.1 Integration

As mentioned before and elaborated below, we will use a finite element discretization. Therefore, we will need to be able to evaluate integration over the material part of each element for accurate forces. However, instead of recalculating a subtesselation of the material region every time the embedding information changes (as done in e.g. [7]), we use the fact that we already have a detailed surface representation within each element from the procedure outlined above. Further, all occurring integrands will be polynomials (or approximated as such). We then apply Gauß' divergence theorem to transform the volume integral to one over the material surface (cf. [58]). This transformation is, of course, not unique; however it is very easy to perform due to the polynomial nature of the integrands. We will demonstrate the idea with the following 3D example and a monomial:

$$f(x,y,z) = x^p y^q z^r \quad \text{with } p,q,r \in \mathbb{N} \tag{2.44}$$

$$\int_{S^\alpha} x^p y^q z^r \, \mathrm{d}\boldsymbol{X} = \int_{\partial S^\alpha} \nabla \cdot \left(\frac{1}{p+1} x^{p+1} y^q z^r, 0, 0 \right) \boldsymbol{n}(\boldsymbol{X}) \, \mathrm{d}\boldsymbol{S}(\boldsymbol{X}). \tag{2.45}$$

We split up any occurring quadrilateral into four triangles by simply adding the midpoint as an additional intersection point. We can additionally perform a change of coordinates to map each such triangle to the standard one defined by $\{(0,0),(1,0),(0,1)\}$. Lastly, we apply a Gaussian quadrature rule on the standard triangle. The weights and quadrature points for these can be found in Table 2.2 (from [38]). In two dimensions, any surface integral is simply mapped to the interval $[0,1]$. The weights and quadrature

points for this simpler case can be found in any text book on numerical analysis. For completeness, we state them in Table 2.1.

order	weights	points	
1	1	$\frac{1}{2}$	$\frac{1}{2}$
3	$\frac{1}{2}$	$\frac{1}{2} - \frac{\sqrt{3}}{3}$	$\frac{1}{2} + \frac{\sqrt{3}}{3}$
	$\frac{1}{2}$	$\frac{1}{2} + \frac{\sqrt{3}}{3}$	$\frac{1}{2} - \frac{\sqrt{3}}{3}$

Table 2.1: Gaussian quadrature weights and points

order	weights	points		
1	1	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
2	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{2}{3}$
	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$
	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{6}$	$\frac{1}{6}$
3	$-\frac{27}{48}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
	$\frac{25}{48}$	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{3}{5}$
	$\frac{25}{48}$	$\frac{1}{5}$	$\frac{3}{5}$	$\frac{1}{5}$
	$\frac{25}{48}$	$\frac{3}{5}$	$\frac{1}{5}$	$\frac{1}{5}$

Table 2.2: Gaussian quadrature weights and points for triangles

With a specific discretization in mind (e.g. as detailed in Section 2.4), all integrands can be predetermined as they will be products of basis functions. By working out the specific cases that can occur, one is able to always apply the quadrature scheme of minimal but appropriate order. Again, since the integrands are polynomials, this results in exact evaluation.

2.4 Finite Element Discretization for Elasticity[1]

In this section we will give an extensive overview about the discretization of elasticity in the framework of the finite element method (FEM) (see also our work in [141], or alternatively [127]). We use linear basis function in order to keep integration simple. Even though our applications will feature both quasistatic as well as dynamic discretization,

[1]Section based on Publication [141]: J.M. Teran, J.L. Hellrung, and J. Hegemann. Simulation of Elasticity, Biomechanics, and Virtual Surgery. *IAS/Park City Mathematics Series*, 2012. To appear.

we will here focus on the latter due to its generality. Furthermore, we discuss a nonlinear constitutive model for the same reason. The case of quasistatic, linear elasticity can be easily deduced from what is presented here, or looked up in the author's Diploma thesis [54], which also contains a detailed derivation of the equilibrium equation. The same applies to the even easier case of Poisson's equation, investigated in Section 3, which is discretized in a completely analogous fashion (alternatively, see [41]).

2.4.1 Governing Equations and Backward Euler

We will quantify the *deformation* of the objects of interest in terms of the mapping $\boldsymbol{\varphi}$ between an initial (or material) $\Omega^0 \in \mathbb{R}^d$ and current (or deformed) $\Omega \in \mathbb{R}^d$ configuration. We will label points in Ω^0 as $\boldsymbol{X}$ and points in Ω as $\boldsymbol{x}$ (see Figure 2.3), which leads to the following notation:

$$\boldsymbol{\varphi}(\cdot, t)\colon \Omega^0 \to \Omega \tag{2.46}$$

$$\boldsymbol{\varphi}(\boldsymbol{X}, t) = \boldsymbol{x}. \tag{2.47}$$

From this, we define the *displacement*

$$\boldsymbol{u}(\cdot, t)\colon \Omega^0 \to \mathbb{R}^d \tag{2.48}$$

$$\boldsymbol{u}(\boldsymbol{X}, t) = \boldsymbol{\varphi}(\boldsymbol{X}, t) - \boldsymbol{X}. \tag{2.49}$$

The *deformation gradient* refers to $\frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{X}}$ and is often denoted with $\boldsymbol{F}$:

$$\boldsymbol{F}(\cdot, t)\colon \Omega^0 \to \mathbb{R}^{d\times d} \tag{2.50}$$

$$\boldsymbol{F} = \frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{X}}. \tag{2.51}$$

We use a hyperelastic idealization of the material response to deformation with a Finite Element Method (FEM) discretization. For hyperelastic materials, the first Piola-Kirchoff stress $\boldsymbol{P}$ is related to an elastic energy density Ψ as ([16])

$$\boldsymbol{P} = \frac{\partial \Psi}{\partial \boldsymbol{F}}, \tag{2.52}$$

where $\boldsymbol{F}$ is the deformation gradient.
(*Remark:* For the case of linear elasticity it is more common to use the Cauchy stress tensor $\boldsymbol{\sigma}$ instead, and we will do so in Chapter 3. Moreover, in that case all deformations are assumed to be small enough such that $\boldsymbol{\sigma} \approx \boldsymbol{P}$, see [63].)

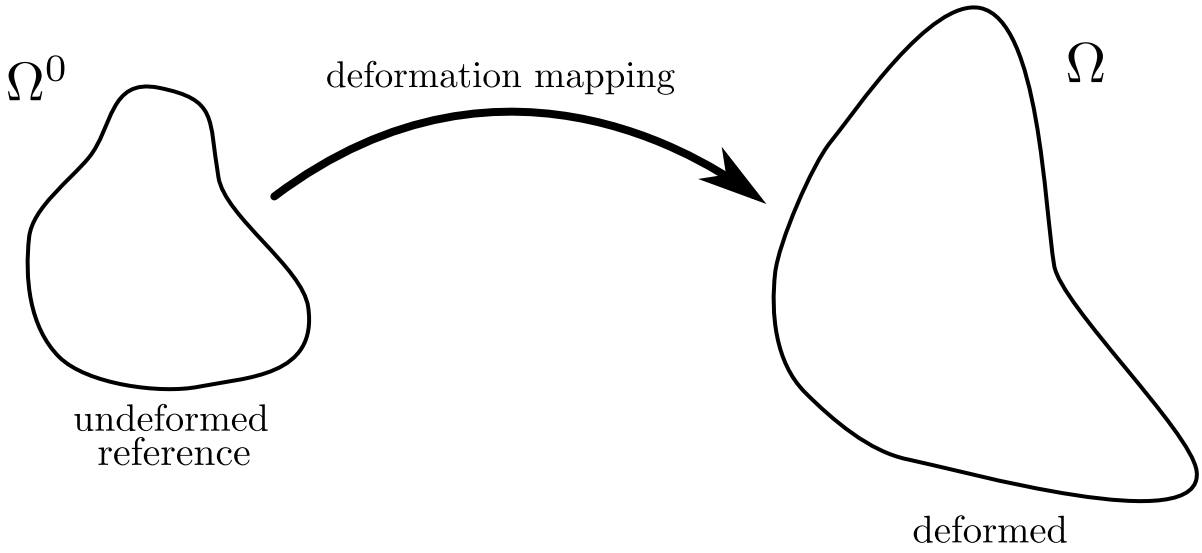


Figure 2.3: We will be solving for the mapping between the current configuration and a rest configuration. Stresses will arise via elasticity to resist changes in shape induced by this motion.

The equations of motion for dynamic elasticity are given by Newton's second law in density form ([52]):

$$\begin{cases} \rho_0 \dfrac{\partial^2 \boldsymbol{u}}{\partial t^2} = \nabla^{\boldsymbol{X}} \cdot \boldsymbol{P}(\boldsymbol{u}) + \boldsymbol{f}^{\text{ext}} & \in \Omega^0 \\ \boldsymbol{u}(\cdot, t) = \boldsymbol{g}(\cdot, t) & \in \partial\Omega_{\text{D}} \\ (\boldsymbol{P} \cdot \hat{\boldsymbol{n}})(\cdot, t) = \boldsymbol{h}(\cdot, t) & \in \partial\Omega_{\text{N}}, \end{cases} \tag{2.53}$$

with ρ_0 being the material density. To allow for larger time steps, we use an implicit backward Euler scheme to update the particle positions. We first introduce an auxiliary variable, $\boldsymbol{v}$ ("velocity"), to transform (2.53) into a first-order system (in time):

$$\frac{\partial \boldsymbol{u}}{\partial t} = \boldsymbol{v}, \tag{2.54}$$

$$\rho_0 \frac{\partial \boldsymbol{v}}{\partial t} = \nabla^{\boldsymbol{X}} \cdot \boldsymbol{P} + \boldsymbol{f}^{\text{ext}}. \tag{2.55}$$

The backward Euler time discretization thus gives

$$\frac{1}{\Delta t}\left(\boldsymbol{u}(\cdot, t+\Delta t) - \boldsymbol{u}(\cdot, t)\right) = \boldsymbol{v}(\cdot, t+\Delta t), \tag{2.56}$$

$$\rho_0 \frac{1}{\Delta t}\left(\boldsymbol{v}(\cdot, t+\Delta t) - \boldsymbol{v}(\cdot, t)\right) = \left(\nabla^{\boldsymbol{X}} \cdot \boldsymbol{P}\right)_{t+\Delta t} + \boldsymbol{f}^{\text{ext}}(\cdot, t+\Delta t). \tag{2.57}$$

Eliminating $\boldsymbol{v}(\cdot, t+\Delta t)$ from (2.57) gives a (non-linear) equation which must be solved for $\boldsymbol{u}(\cdot, t+\Delta t)$ at each time-step. To simplify the notation, we will refer to the unknown $\boldsymbol{u}(\cdot, t+\Delta t)$ as $\boldsymbol{u}$, and to the known variables $\boldsymbol{u}(\cdot, t)$ and $\boldsymbol{v}(\cdot, t)$ as $\boldsymbol{u}_0$ and $\boldsymbol{v}_0$,

respectively. Thus, equation (2.57) is equivalent to

$$\rho_0 \boldsymbol{u} - \Delta t^2 \nabla^{\boldsymbol{X}} \cdot \boldsymbol{P} = \rho_0 \left(\boldsymbol{u}_0 + \Delta t \boldsymbol{v}_0 \right) + \Delta t^2 \boldsymbol{f}^{\text{ext}}, \tag{2.58}$$

where $\nabla^{\boldsymbol{X}} \cdot \boldsymbol{P}$ is evaluated at $t + \Delta t$ (hence depends on the unknown $\boldsymbol{u}$) and $\boldsymbol{f}^{\text{ext}}$ is also evaluated at $t + \Delta t$.

2.4.2 Weak Formulation and FEM Discretization

We now derive the weak formulation of (2.58) to obtain a finite element discretization in space. We multiply both sides with a test function $\boldsymbol{w}$, integrate over Ω^0, apply integration by parts, and simplify the integrals over $\partial\Omega^0$ by stipulating that $\boldsymbol{w} \equiv 0$ on $\partial\Omega_{\text{D}}$. It will be convenient, at this point, to use the following short-hand notation: subscripts will denote coordinates $(1, \ldots, d)$ and (later) superscripts will denote grid vertices; subscripts separated by a comma will be used to indicate a partial derivative with respect to the corresponding coordinate. With these, we can formulate the weak form as

$$\sum_i \left(\rho_0 u_i w_i - \Delta t^2 \sum_j P_{ij,j} w_i \right) = \sum_i \left(\rho_0 \left((u_0)_i + \Delta t (v_0)_i \right) + \Delta t^2 f_i^{\text{ext}} \right) w_i \tag{2.59}$$

$$\int_{\Omega^0} \sum_i \left(\rho_0 u_i w_i - \Delta t^2 \sum_j P_{ij,j} w_i \right) = \int_{\Omega^0} \sum_i \left(\rho_0 \left((u_0)_i + \Delta t (v_0)_i \right) + \Delta t^2 f_i^{\text{ext}} \right) w_i \tag{2.60}$$

$$\begin{aligned} \sum_i \left(\int_{\Omega^0} \rho_0 u_i w_i + \Delta t^2 \sum_j P_{ij} w_{i,j} \right) = \sum_i \Bigg(& \int_{\Omega^0} \left(\rho_0 \left((u_0)_i + \Delta t (v_0)_i \right) + \Delta t^2 f_i^{\text{ext}} \right) w_i \\ & + \Delta t^2 \int_{\partial\Omega_n} h_i w_i \Bigg). \end{aligned} \tag{2.61}$$

For the finite element discretization, we will again assume a mesh of simplices $S^1, \ldots, S^N$ with grid vertices $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^n \in \mathbb{R}^d$ and with S_k^α denoting the kth grid vertex of simplex S^α. Our finite element space will consist of continuous $\mathbb{R}$-valued functions which are affine over each simplex S^α. Our finite element space is the span of the affine linear nodal basis functions $\{N^a\}$. N^a takes the value 1 at $\boldsymbol{X}^a$ and the value 0 at all other grid vertices:

$$N^a(\boldsymbol{X}^b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{else.} \end{cases} \tag{2.62}$$

We then take our test function in the weak formulation (2.61) to be $\boldsymbol{w} = (w_i = \delta_{ij} N^a)_i$, where j ranges over $1, \ldots, d$ and a ranges over $1, \ldots, n$, to obtain dn equations for $\boldsymbol{u}$.

Likewise, we discretize each coordinate of $\boldsymbol{u}$ as $u_i = \sum_b u_i^b N^b$, giving a (non-linear) system of equations for the coefficients $\vec{u} := \left\{u_i^b\right\}$:

$$0 = q_i^a(\vec{u}) := \sum_b \left(\int_{\Omega^0} \rho_0 N^a N^b \right) u_i^b + \Delta t^2 \sum_j \left(\int_{\Omega^0} P_{ij} N_{,j}^a \right) - b_i^a \tag{2.63}$$

with

$$b_i^a := \int_{\Omega^0} \left(\rho_0 \left((u_0)_i + \Delta t (v_0)_i \right) + \Delta t^2 f_i^{\text{ext}} \right) N^a + \Delta t^2 \int_{\partial \Omega_n} h_i N^a. \tag{2.64}$$

We solve the system defined by (2.63) via Newton iterations:

$$\frac{\partial \vec{q}}{\partial \vec{u}}(\vec{u})\, \Delta \vec{u} + \vec{q}(\vec{u}) = 0, \tag{2.65}$$

$$\vec{u} \leftarrow \vec{u} + \Delta \vec{u}. \tag{2.66}$$

2.4.3 Computing the RHS

We will now go through the implementation details of the various computational steps necessary to advance one time step, from time t to time $t + \Delta t$. We start by assembling the right-hand side of the linear equation system defined by (2.65).

Computing $\vec{b}$

The first part of the right-hand side, $\vec{b}$ and its components defined in (2.64), only depends on $\boldsymbol{u}_0$, $\boldsymbol{v}_0$ and $\boldsymbol{f}^{\text{ext}}$, which remain constant throughout the Newton iterations, and thus needs to be computed only once per time step. We will evaluate $\vec{b}$ via an element-based loop, requiring the evaluation of the integrals (for simplicity reasons, we assume homogeneous Neumann boundary conditions, i.e. $\boldsymbol{h} \equiv 0$ here, but the general case can be easily incorporated into our framework). This leaves us with the computation of

$$\int_{S^\alpha} \left(\rho_0 \left((u_0)_i + \Delta t (v_0)_i \right) + \Delta t^2 f_i^{\text{ext}} \right) N^a. \tag{2.67}$$

We assume that the value of $\boldsymbol{f}^{\text{ext}}$ are given at each grid vertex $\boldsymbol{X}^b$ and the value of ρ_0 for each element S^α. Then, we can expand $(u_0)_i = \sum_b (u_0)_i^b N^b$, $(v_0)_i = \sum_b (v_0)_i^b N^b$, and

$f_i^{\text{ext}} = \sum_b f_i^{\text{ext},b} N^b$, yielding

$$\begin{aligned}
&\int_{S^\alpha} \Big(\rho_0\big((u_0)_i + \Delta t (v_0)_i\big) + \Delta t^2 f_i^{\text{ext}}\Big) N^a \\
&\quad = \int_{S^\alpha} \Big(\rho_0^\alpha\big(\sum_b (u_0)_i^b N^b + \Delta t \sum_b (v_0)_i^b N^b\big) + \Delta t^2 \sum_b f_i^{\text{ext},b} N^b\Big) N^a \\
&\quad = \int_{S^\alpha} \sum_b \Big(\rho_0^\alpha\big((u_0)_i^b + \Delta t (v_0)_i^b\big) + \Delta t^2 f_i^{\text{ext},b}\Big) N^a N^b \\
&\quad = \sum_b \Big(\rho_0^\alpha\big((u_0)_i^b + \Delta t (v_0)_i^b\big) + \Delta t^2 f_i^{\text{ext},b}\Big) \int_{S^\alpha} N^a N^b. \qquad (2.68)
\end{aligned}$$

This will be non-zero only for $a, b \in S^\alpha$ due to the local support of the N^a. It thus suffices to evaluate $\int_{S^\alpha} N^a N^b$ for $a, b \in S^\alpha$. Similarly to Subsection 2.3.1, we use a change of coordinates to the standard element, i.e. the triangle $\{(0,0),(1,0),(0,1)\}$ in two dimensions, and in 3D the tetrahedron defined by $\{(0,0,0),(1,0,0),(0,1,0),(0,0,1)\}$. For inside elements, i.e. those which are not intersected by Γ and are therefore entirely filled with material (see the notation of Section 2.3), we can thus compute the integrals as

$$\int_{S^\alpha} N^a N^b = \begin{cases} \dfrac{1}{6}\,\text{area}\,(S^\alpha) & \text{if } a = b, \\ \dfrac{1}{12}\,\text{area}\,(S^\alpha) & \text{if } a \neq b \end{cases} \qquad (2.69)$$

and

$$\int_{S^\alpha} N^a N^b = \begin{cases} \dfrac{1}{10}\,\text{volume}\,(S^\alpha) & \text{if } a = b, \\ \dfrac{1}{20}\,\text{volume}\,(S^\alpha) & \text{if } a \neq b \end{cases} \qquad (2.70)$$

respectively. The volume terms are easily computed from the element coordinates. For cut elements, these unfortunately cannot be precomputed and we have to refer back to Subsection 2.3.1.

Computing $\vec{q}$

The second part of the right-hand side, $\vec{q}$, needs to be updated after every increment to $\vec{u}$ during the Newton steps. As defined in (2.63), its computation requires the evaluation

of the integrals

$$\sum_b \left(\int_{S^\alpha} \rho_0 N^a N^b \right) u_i^b + \Delta t^2 \sum_j \left(\int_{S^\alpha} P_{ij} N^a_{,j} \right) - b_i^a$$
$$= \rho_0^\alpha \sum_b \left(\int_{S^\alpha} N^a N^b \right) u_i^b + \Delta t^2 \sum_j \left(\int_{S^\alpha} P_{ij} N^a_{,j} \right) - b_i^a, \tag{2.71}$$

where we have used the fact that ρ_0 is given element-wise. We already addressed the evaluation of $\int N^a N^b$, so now we will focus on $\int P_{ij} N^a_{,j}$.

Since $\boldsymbol{P} = \boldsymbol{P}(\boldsymbol{F}) = \boldsymbol{P}(\boldsymbol{F}(\boldsymbol{u}))$, we must first compute $\boldsymbol{F}$ from $\boldsymbol{u}$. First, notice from the equations (2.49) and (2.51) that

$$\boldsymbol{F} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}} + \boldsymbol{I}. \tag{2.72}$$

Since

$$\boldsymbol{u} = \sum_a \boldsymbol{u}^a N^a, \tag{2.73}$$

we can thus compute $\boldsymbol{F}$ as

$$\boldsymbol{F} = \sum_a \boldsymbol{u}^a \frac{\partial N^a}{\partial \boldsymbol{X}} + \boldsymbol{I}, \tag{2.74}$$

and the $\frac{\partial N^a}{\partial \boldsymbol{X}}$ are easy to obtain due to the polynomial nature of the N^a.

2.4.4 Matrix Assembly

The most complicated part of the discretization is the evaluation of $\frac{\partial \vec{q}}{\partial \vec{u}}$. We begin by differentiating (2.63) with respect to u_j^b:

$$\frac{\partial q_i^a}{\partial u_j^b} = \delta_{ij} \int_{\Omega^0} \rho_0 N^a N^b + \Delta t^2 \sum_{k,\ell,m} \int_{\Omega^0} \frac{\partial P_{ik}}{\partial F_{\ell m}} \frac{\partial F_{\ell m}}{\partial u_j^b} N^a_{,k}. \tag{2.75}$$

Given that $\boldsymbol{F} = \sum_a \boldsymbol{u}^a \frac{\partial N^a}{\partial \boldsymbol{X}} + \boldsymbol{I}$ (from (2.74)), one can show that $\frac{\partial F_{\ell m}}{\partial u_j^b} = \delta_{j\ell} N^b_{,m}$. Substituting this into the equation above yields

$$\frac{\partial q_i^a}{\partial u_j^b} = \delta_{ij} \int_{\Omega^0} \rho_0 N^a N^b + \Delta t^2 \sum_{k,m} \int_{\Omega^0} \frac{\partial P_{ik}}{\partial F_{jm}} N^a_{,k} N^b_{,m}. \tag{2.76}$$

The computation of the first integral is easily done, as described above, so the challenge remains to evaluate the fourth-order tensor $\frac{\partial P_{ik}}{\partial F_{jm}}$. We follow the formula in [134] (an alternative way can be found in [88]). For a given $\boldsymbol{F}$ with singular value decomposition

(SVD)

$$\boldsymbol{F} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}, \tag{2.77}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are unitary and $\boldsymbol{\Sigma}$ is a diagonal matrix containing the singular values σ_i. Following the above reference, we obtain the second derivative of the energy density in diagonal space, i.e. $\frac{\partial \boldsymbol{P}}{\partial \boldsymbol{F}}(\boldsymbol{\Sigma})$, by simply computing the derivatives of the energy density with respect to the singular values of $\boldsymbol{F}$, a task typically much easier than to compute $\frac{\partial \boldsymbol{P}}{\partial \boldsymbol{F}}(\boldsymbol{F})$ directly. To obtain $\frac{\partial \boldsymbol{P}}{\partial \boldsymbol{F}}(\boldsymbol{F})$ from the quantity $\frac{\partial \boldsymbol{P}}{\partial \boldsymbol{F}}(\boldsymbol{\Sigma})$, we look at the linearization of the stress $\boldsymbol{P}$ around a given $\boldsymbol{F}$:

$$\delta \boldsymbol{P} = \frac{\partial \boldsymbol{P}}{\partial \boldsymbol{F}}(\boldsymbol{F}) : \boldsymbol{\delta F}, \tag{2.78}$$

which can be shown to be ([133, 141])

$$\delta \boldsymbol{P} = \boldsymbol{U}\left(\frac{\partial \boldsymbol{P}}{\partial \boldsymbol{F}}(\boldsymbol{\Sigma}) : \left(\boldsymbol{U}^{\mathrm{T}}\delta \boldsymbol{F}\boldsymbol{V}\right)\right)\boldsymbol{V}^{\mathrm{T}}. \tag{2.79}$$

Let us define $\hat{\boldsymbol{G}} = \frac{\partial \boldsymbol{P}}{\partial \boldsymbol{F}}(\boldsymbol{\Sigma})$ and have a closer look at (2.79) componentwise:

$$\delta P_{ij} = \sum_{p,r} U_{ip}\left(\sum_{s,t} \hat{G}_{prst} \sum_{k,m} U^{\mathrm{T}}_{sk}\delta F_{km} V_{mt}\right) V^{\mathrm{T}}_{rj} \tag{2.80}$$

and with some re-ordering

$$= \sum_{k,m} \delta F_{km} \underbrace{\sum_{p,r,s,t} \hat{G}_{prst} U_{ip} U_{ks} V_{mt} V_{jr}}_{=:G_{ijkm}}. \tag{2.81}$$

By comparing (2.81) with (2.78), we can see that

$$\frac{\partial P_{ik}}{\partial F_{jm}} = G_{ikjm}, \tag{2.82}$$

which we can now utilize for (2.76).

This gives all the information necessary to evaluate $\frac{\partial \vec{q}}{\partial \vec{u}}$ and thus assemble the system to solve for a solution at each iterations of (2.65). As a sanity check, one sees the symmetry when interchanging $a, i \leftrightarrow b, j$. We will handle a specific constitutive model and choice for the energy density Ψ in our application discussed in Chapter 4.

2.5 Solving the Linear System

For the quasistatic problem of Chapter 3, which yields a symmetric, positive definite system of equations and can therefore be solved with a conjugate gradient (CG) method, we again refer to the Diploma thesis [54] for details on properties and implementation. We will instead again focus on the slightly more general case of the dynamic problem, discussed in Section 2.4 and occurring in Chapter 4, which gives rise to a symmetric, indefinite linear system. We use the *minimal residual method* (MINRES) to solve such a system.

The MINRES algorithm is a variant of the conjugate gradient method that still works in the case of indefiniteness. It was introduced in [110] and its convergence behaviour further analysed in [109]. It minimizes the 2-norm of the residual of a linear equation system

$$\boldsymbol{A}\vec{x} = \vec{b} \tag{2.83}$$

with a symmetric, indefinite matrix $\boldsymbol{A}$. An implementation is given by Algorithms 2.2.

The computation of the matrix-vector multiplication $\boldsymbol{A}\vec{v}_k$ in each iteration is the obvious starting point for parallelization, and this is exactly how we modified our implementation. Further, we use a simple Jacobi (or diagonal) preconditioning:

$$\boldsymbol{M}_1^{-1}\boldsymbol{A}\boldsymbol{M}_2^{-1}\vec{y} = \boldsymbol{M}_1^{-1}\vec{b} \tag{2.84}$$

and

$$\vec{x} = \boldsymbol{M}_2^{-1}\vec{y} \tag{2.85}$$

with

$$(M_1)_{ij} = \begin{cases} \operatorname{sign}(A_{ij})\sqrt{|A_{ii}|} & \text{if } i = j \text{ and } A_{ii} \neq 0 \\ 1 & \text{if } i = j \text{ and } A_{ii} = 0 \\ 0 & \text{else} \end{cases} \tag{2.86}$$

$$(M_2)_{ij} = \begin{cases} \sqrt{|A_{ij}|} & \text{if } i = j \text{ and } A_{ii} \neq 0 \\ 1 & \text{if } i = j \text{ and } A_{ii} = 0 \\ 0 & \text{else.} \end{cases} \tag{2.87}$$

The above preconditioning is easy to implement, stable and fast to execute, while yielding decent convergence improvements for our numerical experiments. Note that constructing effective and stable preconditioners for symmetric indefinite systems is in

Algorithm 2.2 The MINRES algorithm.

// Input: sym. matrix $\boldsymbol{A}$, rhs $\vec{b}$, initial guess $\vec{x}_0$, tolerance tol
// Initialization
$\vec{v}_1 \leftarrow \vec{b} - \boldsymbol{A}\vec{x}_0$
$r \leftarrow \|\vec{v}_1\|_2$
$\beta_1 \leftarrow r$; $\eta \leftarrow \beta_1$
$\gamma_0 \leftarrow 1$; $\gamma_1 \leftarrow 1$
$\sigma_0 \leftarrow 0$; $\sigma_1 \leftarrow 0$
$\vec{v}_0 \leftarrow \vec{0}$; $\vec{w}_0 \leftarrow \vec{0}$; $\vec{w}_{-1} \leftarrow \vec{0}$;
for all $k = 1$ **to** max_iter **do**
  // check for convergence
  if $r < tol$ **then**
    break
  end if
  $\vec{v}_k \leftarrow \frac{1}{\beta_k}\vec{v}_k$
  $\alpha_k \leftarrow \vec{v}_k^{\mathrm{T}}\boldsymbol{A}\vec{v}_k$
  $\vec{v}_{k+1} \leftarrow \boldsymbol{A}\vec{v}_k - \alpha_k\vec{v}_k - \beta_k\vec{v}_{k-1}$
  $\beta_{k+1} \leftarrow \|\vec{v}_{k+1}\|_2$
  $\delta \leftarrow \gamma_k\alpha_k - \gamma_{k-1}\sigma_k\beta_k$
  $\rho_1 \leftarrow \sqrt{\delta^2 + \beta_{k+1}^2}$
  $\rho_2 \leftarrow \sigma_k\alpha_k + \gamma_{k-1}\gamma_k\beta_k$
  $\rho_3 \leftarrow \sigma_{k-1}\beta_k$
  $\gamma_{k+1} \leftarrow \frac{\delta}{\rho_1}$
  $\sigma_{k+1} \leftarrow \frac{\beta_{k+1}}{\rho_1}$
  $\vec{w}_k \leftarrow \frac{1}{\rho_1}(\vec{v}_k - \rho_3\vec{w}_{k-2} - \rho_2\vec{w}_{k-1})$
  $\vec{x}_k \leftarrow \vec{x}_{k-1} + \gamma_{k+1}\eta\vec{w}_k$
  $\eta \leftarrow -\sigma_{k+1}\eta$
  $r \leftarrow |\sigma_{k+1}|r$
end for

general still an open problem [145]. For more details on preconditioning in general as well as different techniques we refer to [12, 120] and references therein. For the inverse problem discussed in Chapter 3, we also give an overview on preconditioning that particular class of problems in Section 3.5.

3 Inverse Parameter and Interface Estimation[1]

3.1 Introduction

The identification of coefficients occurring in elliptic partial differential equations (PDEs) is a problem that arises in several different fields, including solid and fluid mechanics, image processing and many more (see e.g. [97]). In the case of medical imaging, inverse parameter estimation could potentially be used to determine the properties and location of different tissue types while using minimally invasive technologies (see [89] and references therein) instead of dissecting the patient. Classifying the elastic properties of tissue and locating abnormalities can help to identify and pinpoint cancerous growth [90]. Further, the elastic properties of bones, muscles and tissue in contact can be approximated as piecewise constant but their actual values may vary from patient to patient. Knowing their exact characteristics can be used to reset, adjust and finetune simulations [121]. This allows for better understanding of the biomechanical configuration at hand and therefore better, individualized treatment for every patient.

In [68], Ito and Kunisch suggested combining an output least squares and equation error formulation with the augmented Lagrangian method to solve inverse parameter estimation problems, an approach that has since become very successful, see e.g. [28, 29, 33, 50, 77, 152]. Usually, these methods discretize the coefficients over the entire domain and use a total variation regularization to achieve an essentially piecewise constant solution for the coefficients. This, and the homogeneous structure of the coefficients in many applications, e.g. geophysical sciences [5, 76], inverse scattering [81] and medical imaging as mentioned before, motivates to include the piecewise constant nature of the coefficients into the approach to these problems.

In addition to estimating the coefficients of a PDE, the geometry of the domain of

[1] Chapter based on Publication [55]: J. Hegemann, A. Cantarero, C. L. Richardson, and J. M. Teran. An explicit update scheme for inverse parameter and interface estimation of piecewise constant coefficients in linear elliptic PDEs. *SIAM Journal on Scientific Computing*, 35(2):A1098-A1119, 2013.

interest is important for many applications, such as the locations of cavities within a material or an optimal shape design under certain exterior conditions. For this so-called inverse geometric problem, where the unknown is a geometric shape, many different approaches have been proposed, e.g. [59, 60, 61, 75, 76]. In [122], Santosa suggested the use of the level set method, developed by Osher and Sethian [107], for inverse obstacle problems. The implicit representation of an interface as the zero level set of a function allows for natural handling of topology changes, such as splitting and merging. Therefore, the level set is capable of evolving towards a solution from almost any initial shape, requiring little or no a priori knowledge. This versatility is highly desirable for solving shape optimization and shape reconstruction problems since the topology of the solution is usually unknown, cf. [21, 22, 23, 24, 25, 40, 69, 108, 114, 115, 116, 132, 143, 144].

Much of the previous work has been done on the model inverse problem arising from Poisson's equation,

$$\begin{cases} -\nabla \cdot (\beta \nabla u) = f & \text{in } \Omega, \\ u = g_1 & \text{on } \partial\Omega_{\mathrm{D}}, \\ \dfrac{\partial u}{\partial n} = g_2 & \text{on } \partial\Omega_{\mathrm{N}}, \end{cases}$$

with applications ranging from electrical impedance tomography [17, 34, 132] and DC resistivity [130] to ground water and oil reservoir investigations [73, 74]. Some authors, e.g. [30, 39, 100, 139], have combined the two aforementioned approaches for the case of Poisson's equation. The goal of these papers is to recover the coefficients as well as the unknown interface.

There has been some work to extend these or similar techniques to the related elliptic inverse problem originating in linear elasticity,

$$\begin{cases} -\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{u}) = \boldsymbol{f} & \text{in } \Omega, \\ \boldsymbol{u} = \boldsymbol{g} & \text{on } \partial\Omega_{\mathrm{D}}, \\ \boldsymbol{\sigma} \cdot \boldsymbol{n} = \boldsymbol{h} & \text{on } \partial\Omega_{\mathrm{N}}, \end{cases}$$

where

$$\begin{aligned} \boldsymbol{\sigma}(\boldsymbol{u}) &= 2\mu\boldsymbol{\epsilon}(\boldsymbol{u}) + \lambda \operatorname{tr}(\boldsymbol{\epsilon}(\boldsymbol{u}))\, \boldsymbol{I}, \\ \boldsymbol{\epsilon}(\boldsymbol{u}) &= \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^{\mathrm{T}}), \end{aligned}$$

with applications in material property determination and inclusion detection [79, 123] as well as design optimization [1, 2, 11, 27, 83, 126, 148] and medical imaging [62, 71]. The goal of these works has been either to estimate the Lamé parameters [48, 49, 70, 102] or

to solve the inverse geometric problem [4, 10].

In our investigation, we also combine the augmented Lagrangian approach with the level set method. As opposed to the aforementioned work of [30], we do not use the whole finite element space to approximate the coefficients in each element combined with a total variation regularization. That approach requires an additional non-linear solve for the coefficients at every grid point at every iteration. The versatility of allowing the coefficients to spatially vary is typically dominated by the TV norm, which leads to smoothing of the solution and to essentially piecewise constant results.

Our approach incorporates the information about the nature of the piecewise constant coefficients directly into the method, which allows us to avoid the additional solve corresponding to the coefficients and instead obtain an explicit, and therefore very efficient, update at each iteration. Furthermore, we obtain the velocity for the level set evolution by use of the shape derivative of our objective functional, leading to an equally efficient update of the interface geometry.

The apparent limitation to truly piecewise constant coefficients is justified in many applications, where the materials are essentially homogeneous. Other authors have successfully included the assumption of piecewise constant coefficients into their approaches [36, 39, 100, 139]. However, our method uses a different objective functional and is more efficient because we use only one linear solve per iteration; all other variables are updated explicitly. For simplicity, we limit our investigations to the example problems of Poisson's equation and linear elasticity, but the methodology is rather general and can be extended to other inverse problems constrained by linear elliptic PDEs.

3.2 Problem Formulation

Let $\Omega \subset \mathbb{R}^d$ be open and bounded, with a smooth or piecewise smooth boundary $\partial\Omega$. We model two elliptic inverse problems: Poisson's equation as well as linear elasticity. Due to the ill-posedness of both inverse problems, we use output-least-squares with an observation $u_0 \in L^2(\Omega)$ of the solution to recover the unknown coefficients occurring in the respective PDE. We allow the coefficients to have jump discontinuities across some unknown interface Γ in Ω and recover Γ as well. This interface Γ, which separates the domain into two disjoint open sets Ω_1 and Ω_2, is assumed to be (piecewise) smooth and is represented implicitly by a level set function ϕ. The subsets Ω_1 and Ω_2 correspond to the regions of positive and negative function values of ϕ, see Figure 3.1:

$$\begin{aligned}\Omega_1 &= \{\boldsymbol{x} \in \Omega | \phi(\boldsymbol{x}) < 0\},\\ \Omega_2 &= \{\boldsymbol{x} \in \Omega | \phi(\boldsymbol{x}) > 0\},\\ \Gamma &= \{\boldsymbol{x} \in \Omega | \phi(\boldsymbol{x}) = 0\} = \partial\Omega_1 \backslash \partial\Omega = \partial\Omega_2 \backslash \partial\Omega.\end{aligned}$$

(The limitation to two regions is for simplicity only; a generalization to arbitrary numbers of subregions is easily obtainable via multiple level sets, see e.g. [30, 138, 147].) We assume the coefficients to be constant in each Ω_i and denote the restriction of a value in one region by a subscript i.

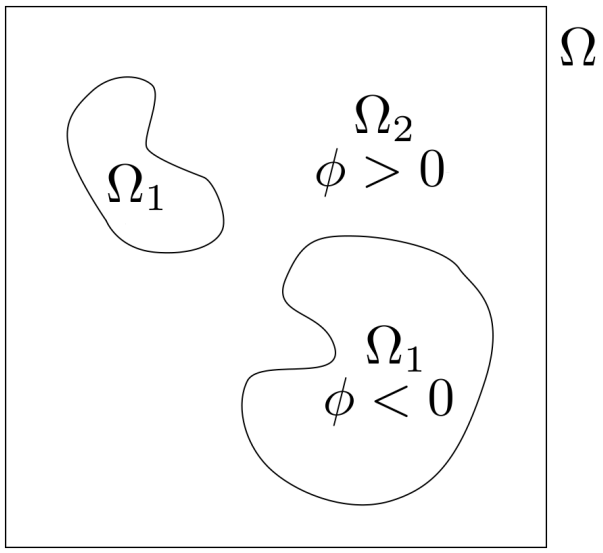


Figure 3.1: Problem setting: Two materials, represented by Ω_1 and Ω_2, with different, unknown properties are in contact. The interface between them, where the parameters have a jump discontinuity, is unknown as well.

3.2.1 Poisson's Equation

We employ Poisson's equation as a first case to show the simplicity of our approach:

$$\begin{cases} -\nabla \cdot (\beta \nabla u) = f & \text{in } \Omega, \\ u = g & \text{on } \partial\Omega_{\mathrm{D}}, \\ \dfrac{\partial u}{\partial n} = 0 & \text{on } \partial\Omega_{\mathrm{N}}, \end{cases} \tag{3.1}$$

with piecewise constant coefficient

$$\beta = \begin{cases} \beta_1 & \text{in } \Omega_1, \\ \beta_2 & \text{in } \Omega_2. \end{cases} \tag{3.2}$$

We solve the constrained minimization problem

$$\min_{\beta_1,\beta_2,\Gamma} \int_\Omega |u - u_0|^2 \, \mathrm{d}x + w|\Gamma| \qquad \text{such that } u \in H^1(\Omega) \text{ is a solution to (3.1)}, \tag{3.3}$$

where $|\Gamma| = \mathcal{H}^{d-1}(\Gamma)$ denotes the $(d-1)$-dimensional Hausdorff measure of Γ and $w > 0$ is a weighting parameter. The corresponding augmented Lagrangian functional for any

$r > 0$ is defined as

$$\mathcal{L}_{\mathrm{A}}(u, v, \beta_1, \beta_2, \Gamma; r, w) = \frac{1}{2}\int_\Omega |u - u_0|^2 \,\mathrm{d}x + \int_\Omega (\nabla v \cdot \beta \nabla u - fv)\,\mathrm{d}x + \frac{r}{2}\int_\Omega |\nabla \cdot \beta \nabla u + f|^2 \,\mathrm{d}x + w|\Gamma|. \tag{3.4}$$

The use of the augmented Lagrangian framework, rather than a regular Lagrangian approach, is motivated by its superior stability properties for iterative schemes, especially in case of noise present in the observation u_0 (see e.g. [28]). We find the saddle point of (3.4) by solving the first order optimality conditions:

$$\frac{\partial \mathcal{L}_{\mathrm{A}}}{\partial u} = u - u_0 - \nabla \cdot \beta \nabla v + r\nabla \cdot \beta \nabla(\nabla \cdot \beta \nabla u + f) \quad = 0, \tag{3.5}$$

$$\frac{\partial \mathcal{L}_{\mathrm{A}}}{\partial \beta_i} = \int_{\Omega_i} \nabla v \cdot \nabla u \,\mathrm{d}x + r\int_{\Omega_i} (\nabla \cdot \beta_i \nabla u + f)\Delta u \;\mathrm{d}x = 0 \qquad \text{for } i = 1, 2. \tag{3.6}$$

For the derivative with respect to the geometry, we use the results about shape derivatives given in Section 2.1. In order to apply these lemmas, we split the integral into the subsets Ω_i, yielding

$$\begin{aligned}\frac{\partial \mathcal{L}_{\mathrm{A}}}{\partial \Omega}[\theta] = \quad & \sum_{i=1}^{2} \int_{\partial\Omega_i \setminus \partial\Omega} \Big(\frac{1}{2}|u - u_0|^2 + \nabla u \cdot \beta_i \nabla v - fv \\ & + \frac{r}{2}|\nabla \cdot \beta_i \nabla u + f|^2\Big)\,\theta(x) \cdot n_i(x)\,\mathrm{d}s + w\int_{\partial\Omega_i \setminus \partial\Omega} \kappa_i(x)\theta(x) \cdot n_i(x)\,\mathrm{d}s \\ = \; & 0,\end{aligned} \tag{3.7}$$

where n_i is the outside normal to Ω_i, and $\kappa_i = \nabla \cdot n_i$ is the mean curvature of $\partial\Omega_i$. Since $n_2 = -n_1$ on $\Gamma = \partial\Omega_1 \cap \partial\Omega_2$ and thus also $\kappa_1 = -\kappa_2 =: \kappa$, we are left with the following equation that needs to vanish on the interface:

$$V = \nabla u \cdot (\beta_1 - \beta_2)\nabla v + w\kappa. \tag{3.8}$$

With A representing the operator $(-\nabla \cdot \beta \nabla)$ and since $A^* = A$ for this problem, we can rewrite (3.5) as

$$(rA^2 + I)u = u_0 - Av + rAf, \tag{3.9}$$

with the boundary conditions specified in (3.1).

3.2.2 Linear Elasticity

We mainly focus on the identification of unknown interfaces and material parameters in the more complicated case of linear elasticity. The governing equations are given by

$$\begin{cases} -\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{u}) = \boldsymbol{f} & \text{in } \Omega, \\ \boldsymbol{u} = \boldsymbol{g} & \text{on } \partial\Omega_{\mathrm{D}}, \\ \boldsymbol{\sigma} \cdot \boldsymbol{n} = \boldsymbol{0} & \text{on } \partial\Omega_{\mathrm{N}}. \end{cases} \tag{3.10}$$

The displacement $\boldsymbol{u}$ and the associated stress $\boldsymbol{\sigma}(\boldsymbol{u})$ are related by Hooke's law,

$$\boldsymbol{\sigma}(\boldsymbol{u}) = \boldsymbol{C} : \boldsymbol{\epsilon}(\boldsymbol{u}) = 2\mu\boldsymbol{\epsilon}(\boldsymbol{u}) + \lambda \operatorname{tr}(\boldsymbol{\epsilon}(\boldsymbol{u}))\, \boldsymbol{I}, \tag{3.11}$$

where C is the fourth order elasticity tensor and the strain is given by

$$\boldsymbol{\epsilon}(\boldsymbol{u}) = \frac{\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^{\mathrm{T}}}{2}. \tag{3.12}$$

The Lamé coefficients

$$\mu = \frac{E}{2(1+\nu)}, \tag{3.13}$$

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \tag{3.14}$$

depend on Young's modulus E and Poisson's ratio ν. We consider a piecewise homogeneous material, and thus μ and λ are piecewise constant and are defined analogously to (3.2). In our approach we solve for μ and λ but use (3.13) and (3.14) to recover the physically meaningful material parameters E and ν.

We have a similar constrained minimization problem,

$$\min_{\mu_1,\lambda_1,\mu_2,\lambda_2,\Gamma} \int_{\Omega} |\boldsymbol{u} - \boldsymbol{u}_0|^2 \, \mathrm{d}\boldsymbol{x} + w|\Gamma| \qquad \text{such that } \boldsymbol{u} \in [H^1(\Omega)]^2 \text{ is a solution to (3.10).} \tag{3.15}$$

Again, we define the corresponding augmented Lagrangian functional for any $r > 0$ as

$$\begin{aligned} \mathcal{L}_{\mathrm{A}}(\boldsymbol{u}, \boldsymbol{v}, \mu_1, \lambda_1, \mu_2, \lambda_2, \Gamma; r, w) = {} & \frac{1}{2} \int_{\Omega} |\boldsymbol{u} - \boldsymbol{u}_0|^2 \, \mathrm{d}\boldsymbol{x}, \\ & + \int_{\Omega} ((2\mu\boldsymbol{\epsilon}(\boldsymbol{u}) + \lambda \operatorname{tr}(\boldsymbol{\epsilon}(\boldsymbol{u}))\, \boldsymbol{I}) : \boldsymbol{\epsilon}(\boldsymbol{v}) - \boldsymbol{f} \cdot \boldsymbol{v}) \, \mathrm{d}\boldsymbol{x}, \\ & + \frac{r}{2} \int_{\Omega} |\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{u}) + \boldsymbol{f}|^2 \, \mathrm{d}\boldsymbol{x} + w|\Gamma|, \end{aligned} \tag{3.16}$$

and compute its optimality conditions,

$$\frac{\partial \mathcal{L}_{\mathrm{A}}}{\partial \boldsymbol{u}} = \boldsymbol{u} - \boldsymbol{u}_0 - \nabla \cdot \boldsymbol{\sigma}(\boldsymbol{v}) + r\nabla \cdot \boldsymbol{\sigma}(\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{u}) + \boldsymbol{f}) = 0 \tag{3.17}$$

and for $i = 1, 2$

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\mathrm{A}}}{\partial \mu_i} &= 2\int_{\Omega_i} \boldsymbol{\epsilon}(\boldsymbol{u}) : \boldsymbol{\epsilon}(\boldsymbol{v}) \,\mathrm{d}\boldsymbol{x} \\
&\quad + 2r\left(\int_{\Omega_i} \nabla \cdot (2\mu_i \boldsymbol{\epsilon}(\boldsymbol{u}) + \lambda_i \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u}))\boldsymbol{I}) \cdot (\nabla \cdot \boldsymbol{\epsilon}(\boldsymbol{u})) \,\mathrm{d}\boldsymbol{x}\right. \\
&\quad \left. + \int_{\Omega_i} \boldsymbol{f} \cdot (\nabla \cdot \boldsymbol{\epsilon}(\boldsymbol{u})) \,\mathrm{d}\boldsymbol{x}\right) \\
&= 0, \\
\frac{\partial \mathcal{L}_{\mathrm{A}}}{\partial \lambda_i} &= \int_{\Omega_i} \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u}))\boldsymbol{I} : \boldsymbol{\epsilon}(\boldsymbol{v}) \,\mathrm{d}\boldsymbol{x} \\
&\quad + r\left(\int_{\Omega_i} \nabla \cdot (2\mu_i \boldsymbol{\epsilon}(\boldsymbol{u}) + \lambda_i \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u}))\boldsymbol{I}) \cdot (\nabla \cdot \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u}))\boldsymbol{I}) \,\mathrm{d}\boldsymbol{x}\right. \\
&\quad \left. + \int_{\Omega_i} \boldsymbol{f} \cdot (\nabla \cdot \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u}))\boldsymbol{I})) \,\mathrm{d}\boldsymbol{x}\right) \\
&= 0.
\end{aligned} \tag{3.18, 3.19}$$

Using the same argument and notation as before, the shape derivative in this case is given by

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\mathrm{A}}}{\partial \Omega}[\theta] = \quad & \sum_{i=1}^{2} \int_{\partial\Omega_i \setminus \partial\Omega} \left(\frac{1}{2}|\boldsymbol{u} - \boldsymbol{u}_0|^2 + \boldsymbol{\epsilon}(\boldsymbol{u}) : \boldsymbol{C}_i : \boldsymbol{\epsilon}(\boldsymbol{v}) - \boldsymbol{f} \cdot \boldsymbol{v}\right. \\
& \left. + \frac{r}{2}|\nabla \cdot \boldsymbol{\sigma}_i(\boldsymbol{u}) + \boldsymbol{f}|^2\right) \theta(\boldsymbol{x}) \cdot \boldsymbol{n}_i(\boldsymbol{x}) \,\mathrm{d}s + w \int_{\partial\Omega_i \setminus \partial\Omega} \kappa_i(\boldsymbol{x}) \theta(\boldsymbol{x}) \cdot \boldsymbol{n}_i(\boldsymbol{x}) \,\mathrm{d}s \\
= 0,
\end{aligned} \tag{3.20}$$

yielding

$$V = \boldsymbol{\epsilon}(\boldsymbol{u}) : (\boldsymbol{C}_1 - \boldsymbol{C}_2) : \boldsymbol{\epsilon}(\boldsymbol{v}) + w\kappa, \tag{3.21}$$

which needs to vanish on the interface.

Again, using the corresponding linear operator $A = \nabla \cdot \boldsymbol{\sigma}(\cdot)$, we obtain $\boldsymbol{u}$ by solving

$$(r\boldsymbol{A}^2 + \boldsymbol{I})\boldsymbol{u} = \boldsymbol{u}_0 - \boldsymbol{A}\boldsymbol{v} + r\boldsymbol{A}\boldsymbol{f} \tag{3.22}$$

with the boundary conditions of the original problem (3.10).

For a more rigorous analysis of augmented Lagrangian methods for identifying discontinuous parameters in elliptic systems, we refer the interested reader to [33, 50]. For a detailed analysis of convergence of level set methods for elliptic inverse problems as well as a more general functional-analytic framework of level set methods and shape reconstruction, we refer to [21, 22, 24].

3.3 Optimization Algorithm

In order to solve the equations derived in the previous section, we use an alternating optimization algorithm. We denote the iterations by n and discretize all operators and functions in an appropriate manner.

3.3.1 Coefficient Updates

For the coefficients, we use the optimality conditions to derive an explicit update rule:

For the case of Poisson's equation we start with the equations in (3.6) at iteration $n+1$ and assume that we already solved for $u = u^{n+1}$ and $v = v^{n+1}$. For a given $i \in \{1, 2\}$, β_i^{n+1} is constant over Ω_i, thus yielding

$$0 = \int_{\Omega_i} \nabla v \cdot \nabla u \, \mathrm{d}x + r \int_{\Omega_i} (\nabla \cdot \beta_i^{n+1} \nabla u + f) \Delta u \, \mathrm{d}x \tag{3.23}$$

$$\Rightarrow \beta_i^{n+1} = \frac{-\int_{\Omega_i} \nabla v \cdot \nabla u \, \mathrm{d}x}{r \int_{\Omega_i} (\Delta u)^2 \, \mathrm{d}x} - \frac{\int_{\Omega_i} f \Delta u \, \mathrm{d}x}{\int_{\Omega_i} (\Delta u)^2 \, \mathrm{d}x} \tag{3.24}$$

since we assume $f \neq 0$ for this case. While ∇u and ∇v can be evaluated numerically, we use a different approach for Δu. Since β_i is constant over Ω_i, we can write $\Delta u = -\frac{f}{\beta_i}$ within each region Ω_i. In this occurrence of the coefficient, we approximate it by its value from the previous iteration, yielding $\Delta u = -\frac{f}{\beta_i^n}$. This allows us to write an explicit update equation:

$$\beta_i^{n+1} = -\frac{\int_{\Omega_i} \nabla v \cdot \nabla u \, \mathrm{d}x}{r/(\beta_i^n)^2 \int_{\Omega_i} f^2 \, \mathrm{d}x} + \beta_i^n \frac{\int_{\Omega_i} f^2 \, \mathrm{d}x}{\int_{\Omega_i} f^2 \, \mathrm{d}x} \tag{3.25}$$

$$= \beta_i^n - \frac{\int_{\Omega_i} \nabla v \cdot \nabla u \, \mathrm{d}x}{r/(\beta_i^n)^2 \int_{\Omega_i} f^2 \, \mathrm{d}x}. \tag{3.26}$$

For linear elasticity we first look at (3.18). Again, we use that the coefficient μ_i^{n+1} is

constant over Ω_i, thus

$$\begin{aligned} 0 = 2 \int_{\Omega_i} \boldsymbol{\epsilon}(\boldsymbol{u}) : \boldsymbol{\epsilon}(\boldsymbol{v}) \, \mathrm{d}\boldsymbol{x} \\ + 2r \left(\int_{\Omega_i} \nabla \cdot (2\mu_i^{n+1} \boldsymbol{\epsilon}(\boldsymbol{u}) + \lambda_i \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u})) \boldsymbol{I}) \cdot (\nabla \cdot \boldsymbol{\epsilon}(\boldsymbol{u})) \, \mathrm{d}\boldsymbol{x} \right. \\ \left. + \int_{\Omega_i} \boldsymbol{f} \cdot (\nabla \cdot \boldsymbol{\epsilon}(\boldsymbol{u})) \, \mathrm{d}\boldsymbol{x} \right) \end{aligned} \tag{3.27}$$

$$\begin{aligned} \Rightarrow 2r\mu_i^{n+1} \int_{\Omega_i} |\nabla \cdot \boldsymbol{\epsilon}(\boldsymbol{u})|^2 \, \mathrm{d}\boldsymbol{x} = r \int_{\Omega_i} -\nabla \cdot (\lambda_i \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u})) \boldsymbol{I} - \boldsymbol{f}) \cdot (\nabla \cdot \boldsymbol{\epsilon}(\boldsymbol{u})) \, \mathrm{d}\boldsymbol{x} \\ - \int_{\Omega_i} \boldsymbol{\epsilon}(\boldsymbol{u}) : \boldsymbol{\epsilon}(\boldsymbol{v}) \, \mathrm{d}\boldsymbol{x} \end{aligned} \tag{3.28}$$

Similarly to before, we use the PDE to substitute $2\mu_i \nabla \cdot \boldsymbol{\epsilon}(\boldsymbol{u}) = -\nabla \cdot \lambda_i \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u})) \boldsymbol{I} - \boldsymbol{f}$ and use $\mu_i = \mu_i^n$ from the previous iteration for this term, allowing us to write

$$2r\mu_i^{n+1} \int_{\Omega_i} |\nabla \cdot \boldsymbol{\epsilon}(\boldsymbol{u})|^2 \, \mathrm{d}\boldsymbol{x} = 2\mu_i^n \int_{\Omega_i} |\nabla \cdot \boldsymbol{\epsilon}(\boldsymbol{u})|^2 \, \mathrm{d}\boldsymbol{x} - \int_{\Omega_i} \boldsymbol{\epsilon}(\boldsymbol{u}) : \boldsymbol{\epsilon}(\boldsymbol{v}) \, \mathrm{d}\boldsymbol{x} \tag{3.29}$$

$$\Rightarrow \mu_i^{n+1} = \mu_i^n - \frac{\int_{\Omega_i} \boldsymbol{\epsilon}(\boldsymbol{u}) : \boldsymbol{\epsilon}(\boldsymbol{v}) \, \mathrm{d}\boldsymbol{x}}{2r \int_{\Omega_i} |\nabla \cdot \boldsymbol{\epsilon}(\boldsymbol{u})|^2 \, \mathrm{d}\boldsymbol{x}}. \tag{3.30}$$

In the same way, (3.19) leads to

$$\begin{aligned} 0 = \int_{\Omega_i} \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u})) \boldsymbol{I} : \boldsymbol{\epsilon}(\boldsymbol{v}) \, \mathrm{d}\boldsymbol{x} \\ + r \left(\int_{\Omega_i} \left(\nabla \cdot (2\mu_i^{n+1} \boldsymbol{\epsilon}(\boldsymbol{u}) + \lambda_i \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u})) \boldsymbol{I}) - \boldsymbol{f} \right) \right. \\ \left. \cdot (\nabla \cdot \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u})) \boldsymbol{I}) \, \mathrm{d}\boldsymbol{x} \right) \end{aligned} \tag{3.31}$$

$$\begin{aligned} \Rightarrow \lambda_i^{n+1} r \int_{\Omega_i} |\nabla \cdot \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u})) \boldsymbol{I}|^2 \, \mathrm{d}\boldsymbol{x} = r \int_{\Omega_i} (-\nabla \cdot (2\mu_i \boldsymbol{\epsilon}(\boldsymbol{u})) - \boldsymbol{f}) \cdot (\nabla \cdot \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u})) \boldsymbol{I}) \, \mathrm{d}\boldsymbol{x} \\ - \int_{\Omega_i} \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u})) \boldsymbol{I} : \boldsymbol{\epsilon}(\boldsymbol{v}) \, \mathrm{d}\boldsymbol{x}. \end{aligned} \tag{3.32}$$

This time, we use $\lambda_i \nabla \cdot \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u})) \boldsymbol{I} = -\nabla \cdot (2\mu_i \boldsymbol{\epsilon}(\boldsymbol{u})) - \boldsymbol{f}$ from the original problem, with $\lambda_i = \lambda_i^n$ for this occurrence:

$$\begin{aligned} \lambda_i^{n+1} r \int_{\Omega_i} |\nabla \cdot \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u})) \boldsymbol{I}|^2 \, \mathrm{d}\boldsymbol{x} = r \int_{\Omega_i} (-\nabla \cdot (2\mu_i \boldsymbol{\epsilon}(\boldsymbol{u})) - \boldsymbol{f}) \cdot (\nabla \cdot \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u})) \boldsymbol{I}) \, \mathrm{d}\boldsymbol{x} \\ - \int_{\Omega_i} \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u})) \boldsymbol{I} : \boldsymbol{\epsilon}(\boldsymbol{v}) \, \mathrm{d}\boldsymbol{x} \end{aligned} \tag{3.33}$$

$$\Rightarrow \lambda_i^{n+1} = \lambda_i^n - \frac{\int_{\Omega_i} \operatorname{tr}(\boldsymbol{\epsilon}(\boldsymbol{u}))\boldsymbol{I} : \boldsymbol{\epsilon}(\boldsymbol{v}) \, \mathrm{d}\boldsymbol{x}}{r \int_{\Omega_i} |\nabla \cdot \operatorname{tr}(\boldsymbol{\epsilon}(\boldsymbol{u}))\boldsymbol{I}|^2 \, \mathrm{d}\boldsymbol{x}}. \tag{3.34}$$

3.3.2 Interface Update

As mentioned in Sections 2.2 and 3.2, we represent the interface Γ implicitly by the zero contour of a level set function ϕ. The interface is advected according to the level set equation (2.5),

$$\frac{\partial \phi}{\partial t} - \delta_\epsilon(\phi) V = 0,$$

and we obtain the speed V as given by (3.8) and (3.21) respectively.

The curvature term $w\kappa = w \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|}$ in both velocities is discretized with the semi-implicit finite difference scheme of Vese and Chan in [147]. The level set values are defined on a regular Cartesian grid (consisting of the nodes of our triangle mesh); and we will use multi-index notation to identify the grid nodes and multi-index subscripts to indicate function values at the corresponding grid nodes. For a grid spacing $h = \Delta x = \Delta y$, define

$$C_1 = \frac{1}{\sqrt{\left(\frac{\phi^n_{i+1,j} - \phi^n_{i,j}}{h}\right)^2 + \left(\frac{\phi^n_{i,j+1} - \phi^n_{i,j-1}}{h}\right)^2}}, \tag{3.35}$$

$$C_2 = \frac{1}{\sqrt{\left(\frac{\phi^n_{i,j} - \phi^n_{i-1,j}}{h}\right)^2 + \left(\frac{\phi^n_{i-1,j+1} - \phi^n_{i-1,j-1}}{h}\right)^2}}, \tag{3.36}$$

$$C_3 = \frac{1}{\sqrt{\left(\frac{\phi^n_{i+1,j} - \phi^n_{i-1,j}}{h}\right)^2 + \left(\frac{\phi^n_{i,j+1} - \phi^n_{i,j}}{h}\right)^2}}, \tag{3.37}$$

$$C_4 = \frac{1}{\sqrt{\left(\frac{\phi^n_{i+1,j-1} - \phi^n_{i-1,j-1}}{h}\right)^2 + \left(\frac{\phi^n_{i,j} - \phi^n_{i,j-1}}{h}\right)^2}}, \tag{3.38}$$

as well as

$$C = 1 + m(C_1 + C_2 + C_3 + C_4), \tag{3.39}$$

$$m = w \, \frac{\Delta s}{h^2} \delta_\epsilon(\phi^n_{i,j}), \tag{3.40}$$

with δ_ϵ as defined in (2.5). The update steps are then given as

$$\phi_{i,j}^{n+1} = \frac{1}{C}\left[\phi_{i,j}^n + m\left(C_1\phi_{i+1,j}^n + C_2\phi_{i-1,j}^n + C_3\phi_{i,j+1}^n + C_4\phi_{i,j-1}^n\right)\right.$$
$$\left. +\Delta s\,\delta_\epsilon(\phi_{i,j}^n)\,(\nabla u\cdot(\beta_1-\beta_2)\nabla v)_{i,j}\right] \tag{3.41}$$

for Poisson's equation, and

$$\phi_{i,j}^{n+1} = \frac{1}{C}\left[\phi_{i,j}^n + m\left(C_1\phi_{i+1,j}^n + C_2\phi_{i-1,j}^n + C_3\phi_{i,j+1}^n + C_4\phi_{i,j-1}^n\right)\right.$$
$$\left. +\Delta s\,\delta_\epsilon(\phi_{i,j}^n)\,(\boldsymbol{\epsilon}(\boldsymbol{u}):(\boldsymbol{C}_1-\boldsymbol{C}_2):\boldsymbol{\epsilon}(\boldsymbol{v}))_{i,j}\right] \tag{3.42}$$

in the case of linear elasticity.

The level set evolution then reaches its steady state once the optimality condition of the Lagrangian functional is fulfilled. One important detail for the evolution here is that we have to be careful not to reinitialize the level set function ϕ too often. Our inverse problems are ill-posed and therefore, the evolution is very slow and it might take a long time for a nodal value to change sign. By reinitializing too often, these small changes could be revoked. In our method, we reinitialize either when a certain percentage of nodal values of ϕ have changed signs or after a fixed number of iterations have passed since the last reinitialization. A different strategy is to reinitialize after the L_2-norm of ϕ has changed more than a given percentage.

3.3.3 Algorithm

We now outline our algorithm. First, we initialize ϕ^0 with the initial guess for the interface, the coefficients (β_i^0 or μ_i^0, λ_i^0, $(i = 1, 2)$ respectively) with positive values and v^0 as a zero-vector. We also choose a fixed number of iterations *reinit_number* and a percentage *reinit_ratio* for the level set reinitialization condition, as well as initialize the helper variables *number_sign_changes* and *iterations_since_reinit* as zero. Algorithm 3.1 provides the pseudo-code for our method.

Remarks:

- The update order for the different variables is not important. Given an order, it is most efficient to use the most recent values on hand to update the remaining variables.
- Note that $(rA^2 + I)$ never has to be built explicitly; only its application is needed, which makes computations efficient. The matrix $(rA^2 + I)$ is obviously symmetric and positive definite, and therefore we can use a (preconditioned) conjugate gradient method or similar to solve the equation to obtain u. We need only one solve

Algorithm 3.1 An explicit update scheme for inverse parameter and interface estimation of piecewise constant coefficients in linear elliptic PDEs.

Initialization
for all n **do**
  Solve $(rA^2 + I)u^{n+1} = u_0 - Av^n + rAf$
  $v^{n+1} \leftarrow v^n + r(Au^{n+1} - f)$ // explicit update of the Lagrange multiplier
  for all $i = 1$ **to** 2 **do** // explicit update of the coefficients
    // in case of Poisson's equation
    $\beta_i^{n+1} \leftarrow \beta_i^n - \frac{\int_{\Omega_i} \nabla v^{n+1} \cdot \nabla u^{n+1}}{r/(\beta_i^n)^2 \int_{\Omega_i} f^2}$
    or // in case of linear elasticity
    $\mu_i^{n+1} \leftarrow \mu_i^n - \frac{\int_{\Omega_i} \boldsymbol{\epsilon}(\boldsymbol{u}^{n+1}) : \boldsymbol{\epsilon}(\boldsymbol{v}^{n+1})}{2r \int_{\Omega_i} |\nabla \cdot \boldsymbol{\epsilon}(\boldsymbol{u}^{n+1})|^2}$
    $\lambda_i^{n+1} \leftarrow \lambda_i^n - \frac{\int_{\Omega_i} \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u}^{n+1})) I : \boldsymbol{\epsilon}(\boldsymbol{v}^{n+1})}{r \int_{\Omega_i} |\nabla \cdot \mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u}^{n+1})) \boldsymbol{I}|^2}$
  end for
  $\phi^{n+1} \leftarrow \mathrm{Advect}(\phi^n)$ // as described in Section 3.3.2
  for all $i = 1$ **to** *number_nodes* **do** // check ϕ for a sign changes
    if $\phi_i^n \cdot \phi_i^{n+1} < 0$ **then**
      $number\_sign\_changes \leftarrow number\_sign\_changes + 1$
    end if
  end for
  if $iterations\_since\_reinit > reinit\_number$
  or $number\_sign\_changes/number\_nodes > reinit\_ratio$ **then**
    Fast_sweep(ϕ^{n+1}) // reinitialization as a signed distance function
    $number\_sign\_changes \leftarrow 0$
    $iterations\_since\_reinit \leftarrow 0$
  end if
  Check convergence based on
  $E_i \leftarrow \frac{\mu_i(3\lambda_i + 2\mu_i)}{\lambda_i + \mu_i}$
  $\nu_i \leftarrow \frac{\lambda_i}{2(\lambda_i + \mu_i)}$
  and the interface
end for

per iteration since both the Lagrange multiplier v and the coefficients are updated explicitly.

3.4 Numerical Examples

For all examples we use a regular triangle grid of the domain $\Omega = [0,1]^2$ with uniform spacing $\Delta x = \Delta y = h = 1/256$. The PDE solves for u are completed using FEM. The time step of the level set evolution is fixed to $\Delta t = h$ and coincides with a minimizing step of the Lagrangian functional $\mathcal{L}_{\mathrm{A}}$. For the approximation of the interface in (2.5), we set $\epsilon = h$. We reinitialize ϕ either when 10% of the vertices have changed sign or when 100 iterations have passed since the last reinitialization. The convergence is often very fast in the beginning of the computation and then slows down after an initial phase. This accounts for the high total number of iterations we observe and is typical for augmented Lagrangian methods, cf. [30].

3.4.1 Poisson's Equation

We demonstrate our algorithm for the case of Poisson's equation using an example where Ω_1 is composed of two different shapes, a square and an ellipse, included in Ω. We choose the simple case of $f = 1$ and homogeneous boundary conditions. The exact solution is computed with $\beta_1 = 5.0$ and $\beta_2 = 1.0$ and we choose $r = 1.0$ and $w = 5 \cdot 10^{-6}$ and obtain a very accurate recovery of the coefficients and the interface, see Table 3.1 and Figure 3.2.

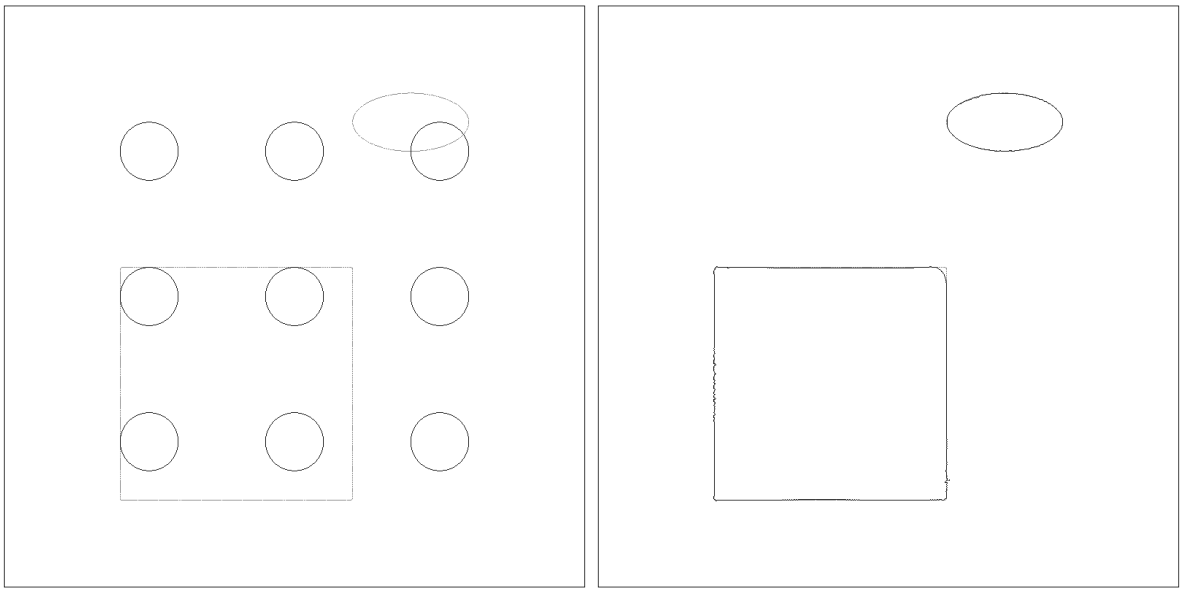


Figure 3.2: Interface recovery for Poisson's equation: exact solution with initial (*left*) and final configuration (*right*).

coefficient	exact	recovered	relative error
β_1	5.0	5.00252	0.0504%
β_2	1.0	1.00089	0.089%

Table 3.1: Recovered coefficients for Poisson's equation.

3.4.2 Linear Elasticity

We focus our numerical experiments on examples in linear elasticity and present multiple settings: we use the same geometric setting as before, with and without the addition of uniformly distributed noise, and also employ a more elaborate geometry. For all these examples, f is set equal to zero.

Square and ellipse

For this example, we choose the same geometric setting as above, with Ω_1 being a square and an ellipse. The boundary conditions are given as pure stretching by $\boldsymbol{u} = (-0.01, 0)$ on the left and $\boldsymbol{u} = (0.01, 0)$ on the right, and free on top and bottom (see Figure 3.3 for an illustration). The material parameters are set to $E_1 = 200.0$, $\nu_1 = 0.28$ and $E_2 = 117.0$, $\nu_2 = 0.33$, for which we choose the initial guesses $E_1 = E_2 = 150.0$, $\nu_1 = \nu_2 = 0.30$. We choose $r = 0.001$, $w = 10^{-6}$ and are able to recover the coefficients (see Table 3.2) and the interface (see Figure 3.4 for the evolution) very accurately for this case.

coefficient	exact	recovered	relative error
E_1	200.0	200.068	0.034%
ν_1	0.28	0.280609	0.2175%
E_2	117.0	117.547	0.4675%
ν_2	0.33	0.329976	0.0073%

Table 3.2: Recovered coefficients for linear elasticity example "*square and ellipse*".

Square and ellipse with noise

We now use the same setting as described above and add noise to the exact solution. We obtain our observation u_0 as follows:

$$\boldsymbol{u}_0 = \boldsymbol{u}^* + \sigma \frac{||\boldsymbol{u}^*||_{L^2}}{||\boldsymbol{S}||_{L^2}} \boldsymbol{S},$$

where u^* is the exact solution for a given set of coefficients. The vector $\boldsymbol{S}$ contains nodal values $s_i \in \mathbb{R}^d$, with $(s_i)_j \in [-1, 1]$ uniformly random, and σ controls the noise level. See

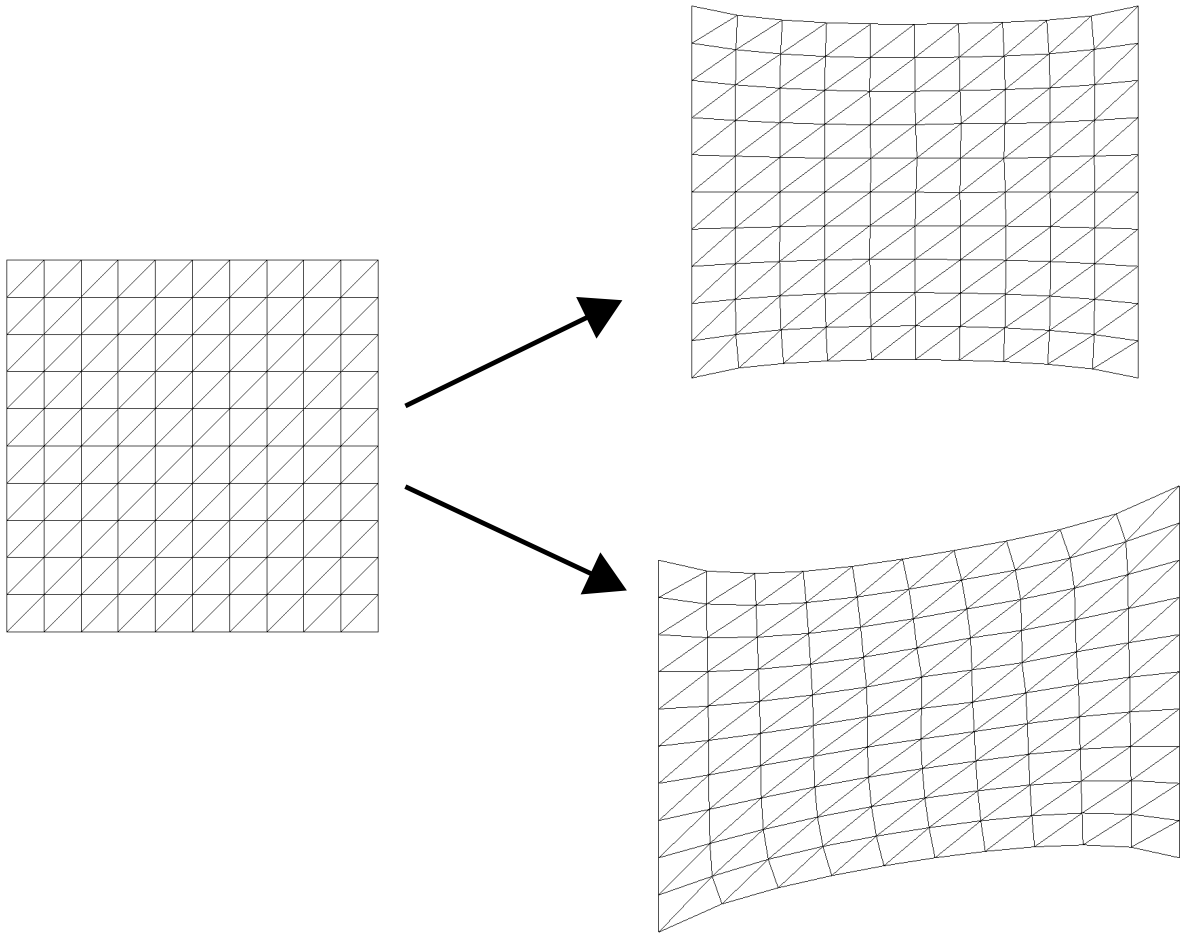


Figure 3.3: Illustration of the boundary displacements used in the examples: the undeformed domain on the *left* is deformed into the configuration on the *right*, either by *pure stretching* in the example "*square and ellipse*" (*upper right*) or by *pulling and shearing* in the example "*elaborate geometry*" (*lower right*). Note that the boundary displacements used in this picture are exaggerated for illustration purposes, the displacements used in the experiments are much smaller.

Table 3.3 and Figure 3.5 for the results with $\sigma = 5\%, w = 10^{-5}$ and $r = 0.005$. Table 3.4 and Figure 3.6 show the results for $\sigma = 10\%, w = 2.0 \cdot 10^{-5}$ and $r = 0.005$, and the results of Table 3.5 and Figure 3.7 were computed with $\sigma = 20\%, w = 2.0 \cdot 10^{-5}$ and $r = 0.0075$,

coefficient	exact	recovered	relative error
E_1	200.0	190.139	4.9305%
ν_1	0.28	0.280381	0.1361%
E_2	117.0	111.176	4.9778%
ν_2	0.33	0.330018	0.0055%

Table 3.3: Recovered coefficients for linear elasticity example "*square and ellipse*", with $\sigma = 5\%$ noise.

coefficient	exact	recovered	relative error
E_1	200.0	221.334	10.6670%
ν_1	0.28	0.278046	0.6979%
E_2	117.0	129.226	10.4957%
ν_2	0.33	0.329707	0.0888%

Table 3.4: Recovered coefficients for linear elasticity example "*square and ellipse*", with $\sigma = 10\%$ noise.

coefficient	exact	recovered	relative error
E_1	200.0	233.301	16.6505%
ν_1	0.28	0.274761	1.8711%
E_2	117.0	134.272	14.7624%
ν_2	0.33	0.330961	0.2912%

Table 3.5: Recovered coefficients for linear elasticity example "*square and ellipse*", with $\sigma = 20\%$ noise.

Elaborate geometry

In our last example, we show that we can target a more complicated geometry as shown in Figure 3.8 on the *left*. The material properties are given as in the previous example, but we change the initial guesses to $E_1 = 230.0$, $\nu_1 = 0.30$ and $E_2 = 80.0$, $\nu_2 = 0.35$, the boundary conditions are specified as $u = (-0.02, -0.01)$ on the left and $u = (0.02, 0.01)$ on the right, and free on top and bottom, i.e. we pull and shear the material (see Figure 3.3 for an illustration). For this example, we set $w = 10^{-5}$ and $r = 0.004$. The results can be seen in Table 3.6 and Figure 3.8 and show how accurately we can recover coefficients and interface even in this setting.

coefficient	exact	recovered	relative error
E_1	200.0	207.974	3.9870%
ν_1	0.28	0.281306	0.4664%
E_2	117.0	121.568	3.9042%
ν_2	0.33	0.330681	0.2063%

Table 3.6: Recovered coefficients for linear elasticity example "*elaborate geometry*".

Elaborate geometry with noise

Just as before, we add different noise levels to this geometry. The results for $\sigma = 5\%$ (with $w = 10^{-4}$ and $r = 0.005$), $\sigma = 10\%$ (with $w = 2.0 \cdot 10^{-4}$ and $r = 0.005$), and $\sigma = 20\%$ (with $w = 2.0 \cdot 10^{-4}$ and $r = 0.005$) are presented in Tables 3.7 - 3.9 and

Figures 3.9 - 3.11.

coefficient	exact	recovered	relative error
E_1	200.0	211.515	5.7575%
ν_1	0.28	0.279285	0.2554%
E_2	117.0	123.206	5.3043%
ν_2	0.33	0.330886	0.2685%

Table 3.7: Recovered coefficients for linear elasticity example "*elaborate geometry*", with $\sigma =$ 5% noise.

coefficient	exact	recovered	relative error
E_1	200.0	214.159	7.0795%
ν_1	0.28	0.271017	3.2082%
E_2	117.0	123.369	5.4436%
ν_2	0.33	0.33157	0.4758%

Table 3.8: Recovered coefficients for linear elasticity example "*elaborate geometry*", with $\sigma =$ 10% noise.

coefficient	exact	recovered	relative error
E_1	200.0	207.135	3.5675%
ν_1	0.28	0.268389	4.1468%
E_2	117.0	116.33	0.5727%
ν_2	0.33	0.330973	0.2948%

Table 3.9: Recovered coefficients for linear elasticity example "*elaborate geometry*", with $\sigma =$ 20% noise.

3.4.3 Parameter Choices

Lastly, we explain the effects of our choices for the parameters r and w on the results presented here. It is well known that the weight on the interface length, w, controls its smoothness: high values can lead to "over-smoothed" interface and even loss of smaller objects (see Figure 3.12 on the left), small values can lead to jagged contours and small noise artefacts can occur (see Figure 3.12 on the right). The augmented weight r needs to be large enough to ensure convergence, especially in the presence of noise. Choosing a larger r is always possible and can lead to better convergence in terms of total number of iterations, but at the price of higher computational cost per iteration due to its roll in Equation (3.9) and (3.22) respectively. In fact, for most examples shown in this section

we used very similar r-values for simplicity reasons and to demonstrate our ability to handle different observations with the same parameter.

3.5 Conclusion

We have presented a fast, accurate and easy to implement algorithm to solve for the arising coefficients as well as the discontinuity interface in elliptic PDEs, demonstrating it for the problems of Poisson's equation and linear elasticity. The advantage of our approach is the treatment of the coefficients as truly piecewise constant. This provides us with a fast explicit update at every iteration since we only need one linear solve at every iteration, all other variables are updated explicitly and we do not need an additional non-linear solve. Further, our approach is general and can be employed to any linear elliptic inverse problem. As presented in this paper, this allows us to provide the simultaneous recovery of coefficients and interface for Poisson's equation and linear elasticity. For the case of Poisson's equation we have to assume a non-zero right hand side but for linear elasticity the forcing term can be of any nature. Several numerical examples show that our method is numerically stable and can handle not only a variety of geometries but also a wide range of noise.

The essential difference of our method compared to previous work in the literature, e.g. [28, 29, 30], is that we reduce the computational cost per iteration from a non-linear solve, originating from the TV regularization, to an explicit update equation for the coefficients. Further, our methods seems to be able to tolerate much higher noise levels than reported from those methods, especially for non-trivial interfaces. Some authors, like [69], can also deal with high observation error, however, they only solve for the interface and assume the constant coefficients to be known a priori.

Due to the ability of our method to handle input data polluted with noise, we expect it to work with experimental data. In the case of medical imaging, the interior measurement u_0 could be acquired by techniques involving ultrasound or MRI, see [71, 75, 89, 90, 121] and references therein.

Currently, our method is based on a first order accurate solver on a triangular mesh. The incorporation of a second order method (cf. [8, 80, 146]) could potentially enhance every iteration. Also, a more advanced preconditioner, possibly based on multigrid approaches (see [87] and the references therein) or especially designed for saddle point problems with PDE constraint [13, 14, 18, 118, 124], could further improve the runtime performance.

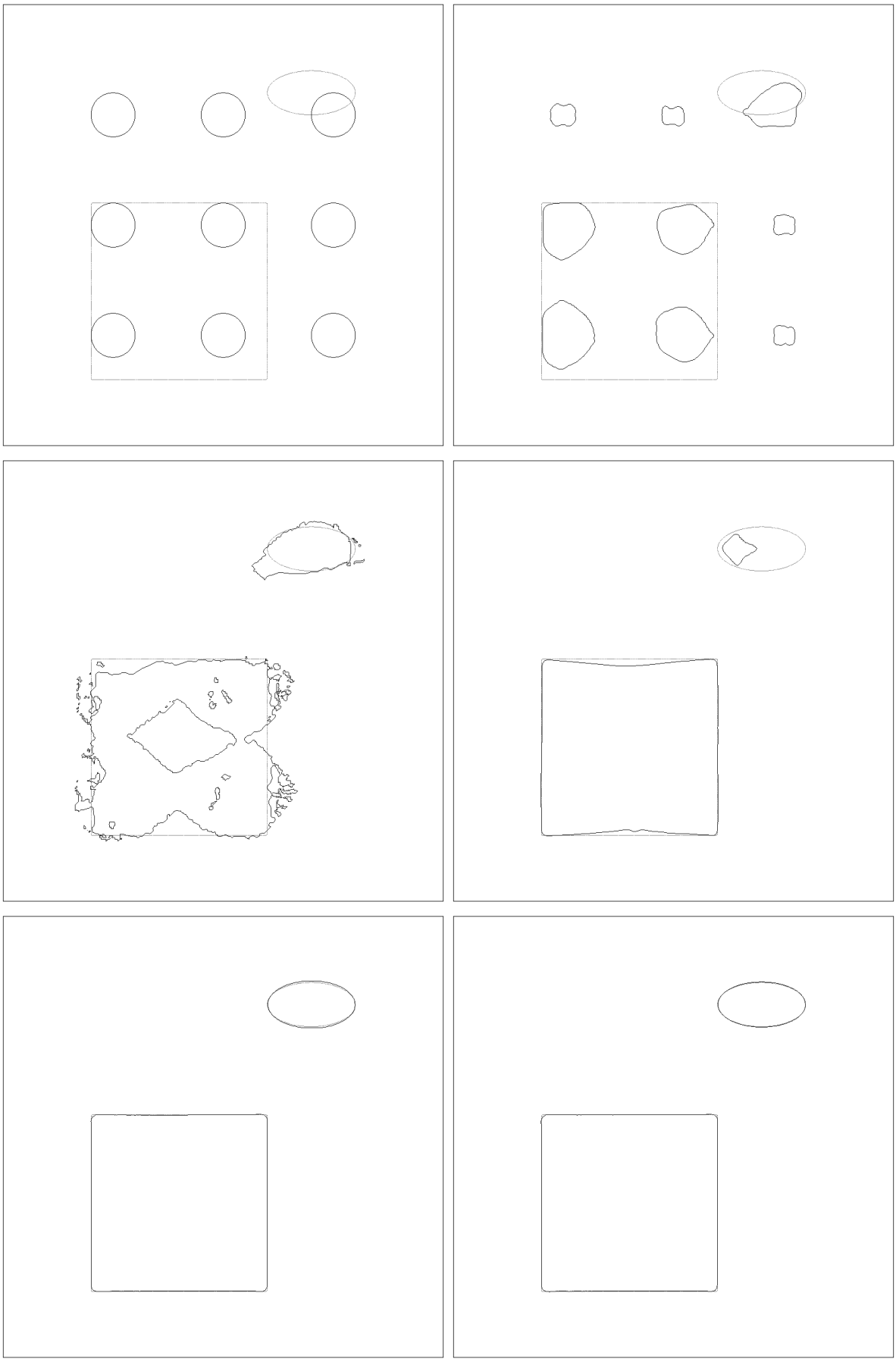


Figure 3.4: Interface evolution for linear elasticity example "*square and ellipse*": after 0, 1000, 2000, 5000, 10000, 12000 iterations (from *top left* to *bottom right*).

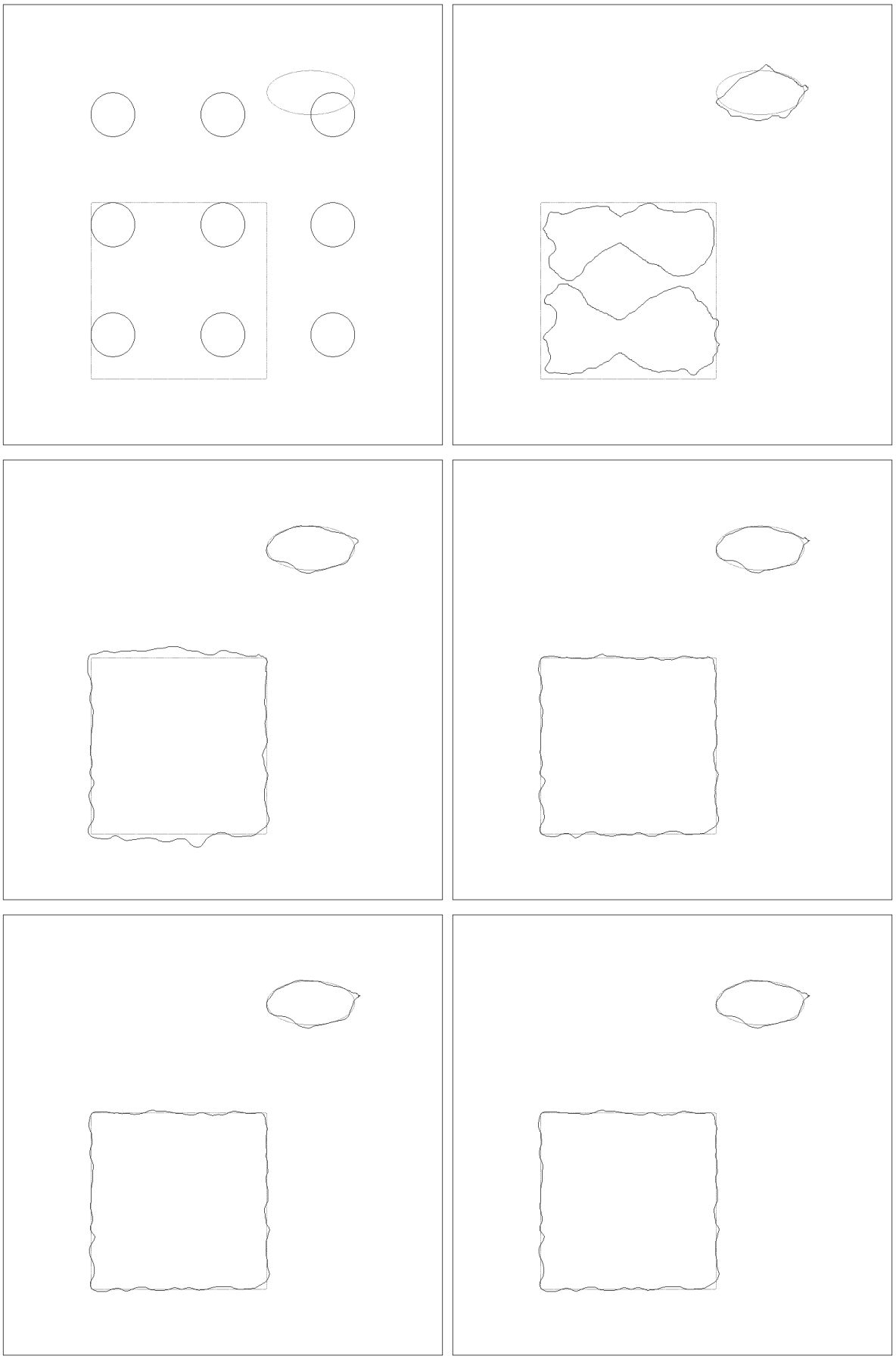


Figure 3.5: Interface evolution for linear elasticity example "*square and ellipse*", with $\sigma = 5\%$ noise: after 0, 1000, 3000, 5000, 10000, 13000 iterations (from *top left* to *bottom right*).

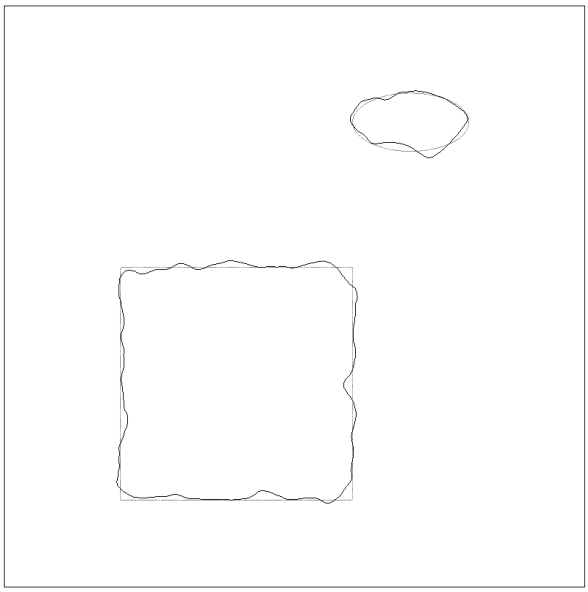


Figure 3.6: Recovered interface for linear elasticity example "*square and ellipse*", with $\sigma =$ 10% noise.

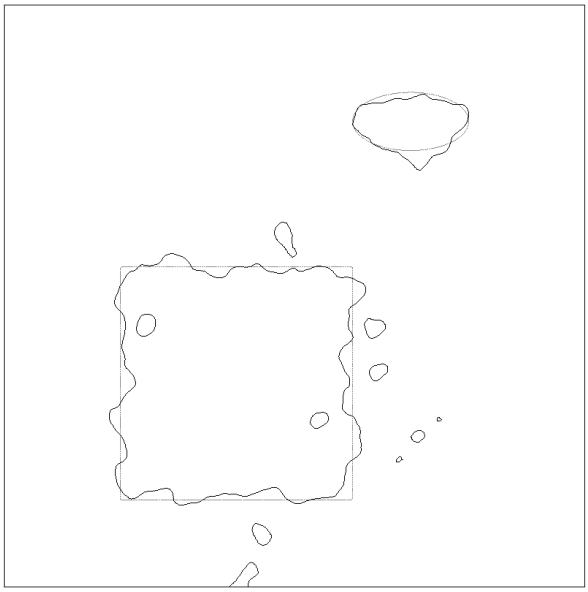


Figure 3.7: Recovered interface for linear elasticity example "*square and ellipse*", with $\sigma =$ 20% noise.

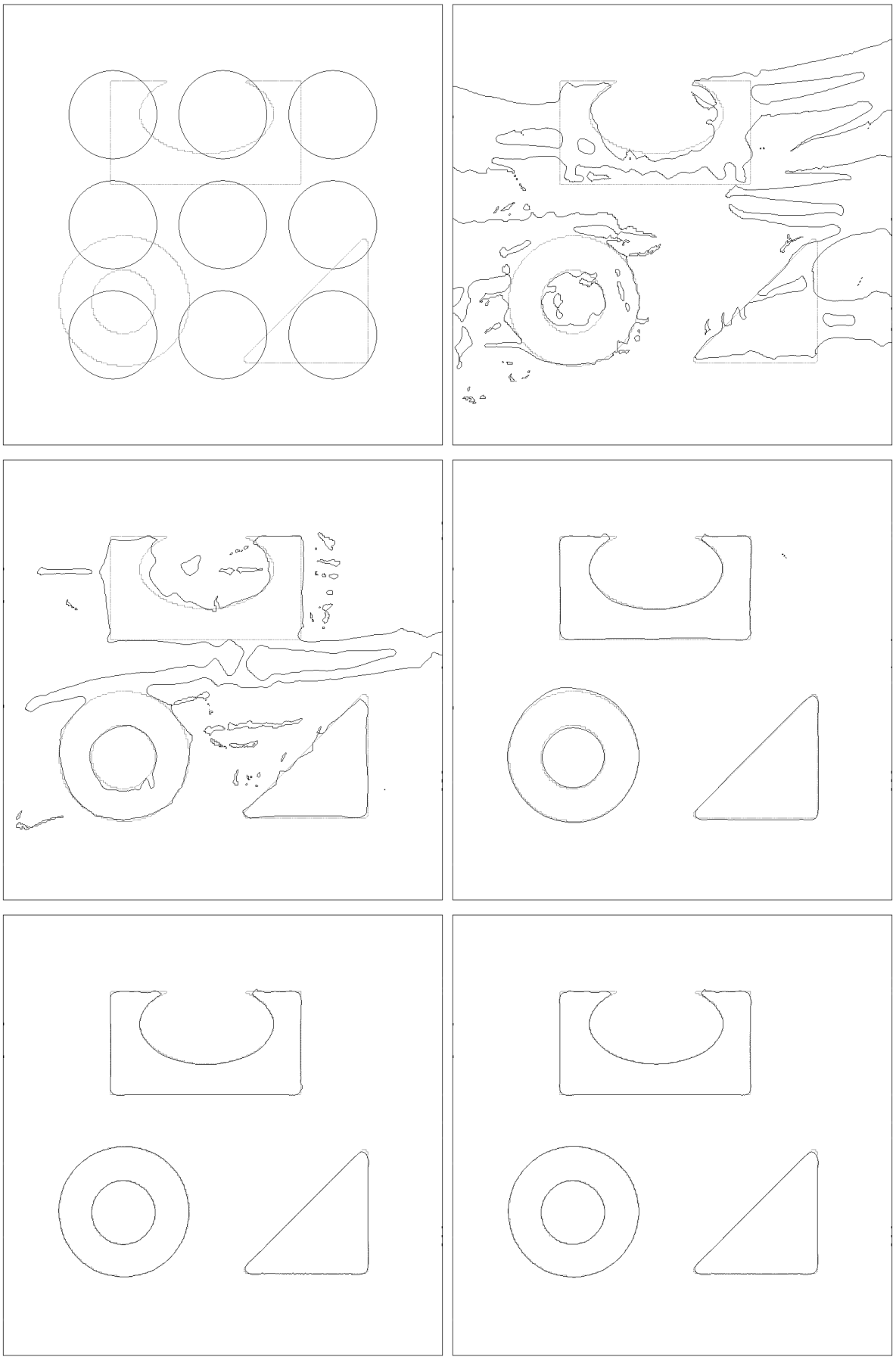


Figure 3.8: Interface evolution for linear elasticity example "*elaborate geometry*": after 0, 1000, 2000, 3000, 4000, 5000 iterations (from *top left* to *bottom right*).

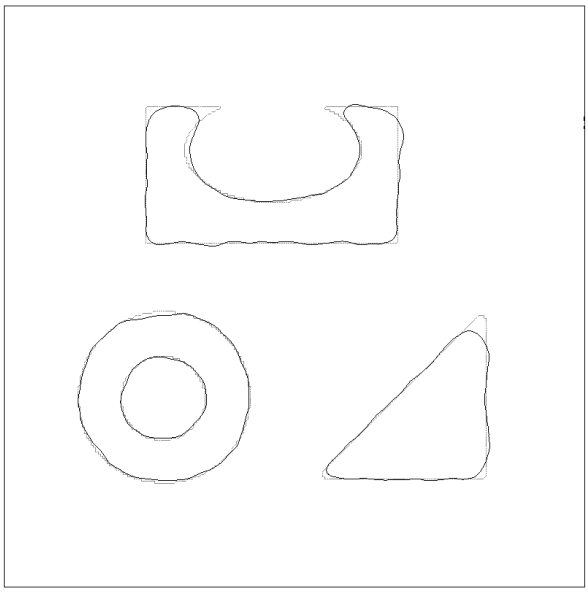


Figure 3.9: Recovered interface for linear elasticity example '*elaborate geometry*", with $\sigma = 5\%$ noise.

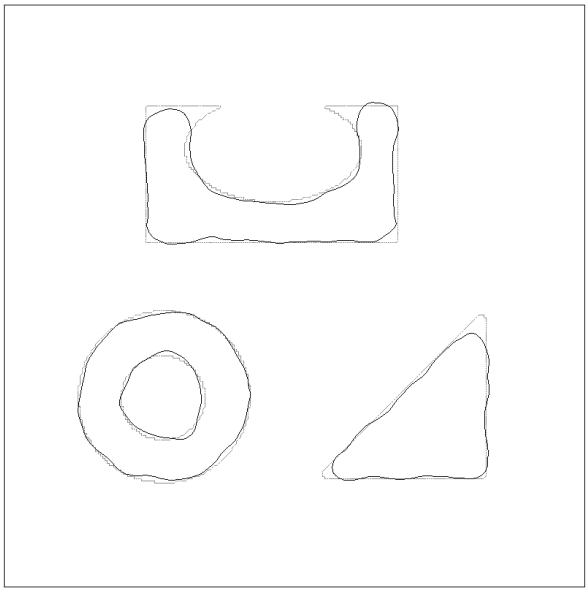


Figure 3.10: Recovered interface for linear elasticity example "*elaborate geometry*", with $\sigma = 10\%$ noise.

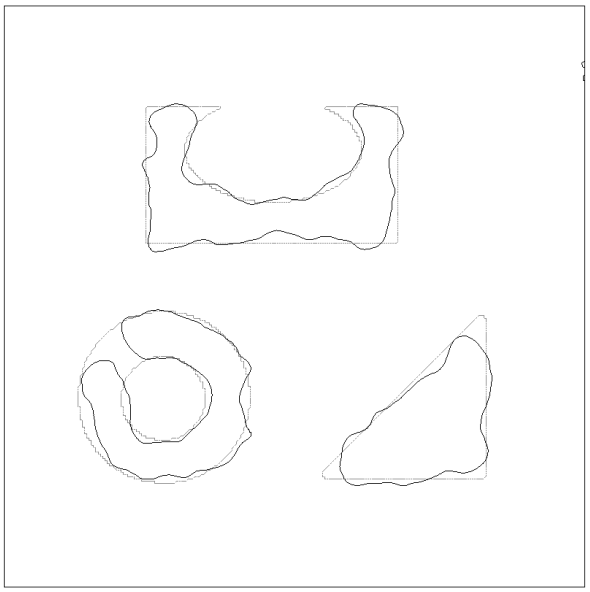


Figure 3.11: Recovered interface for linear elasticity example "*elaborate geometry*", with $\sigma =$ 20% noise.

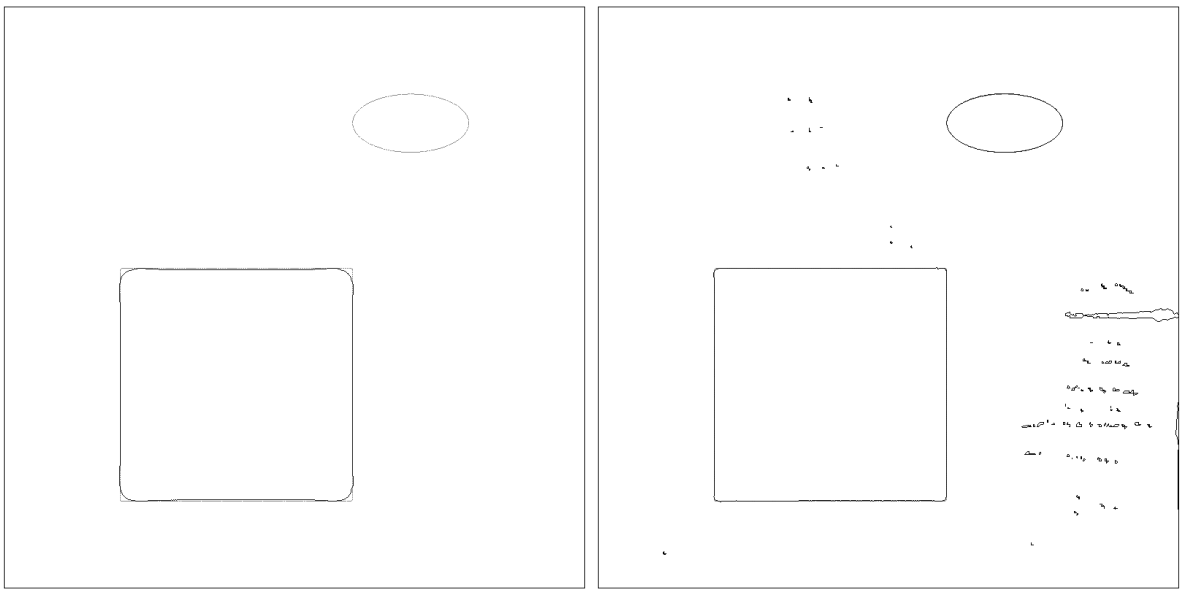


Figure 3.12: Interfaces for linear elasticity example "*square and ellipse*" and different w-values: *on the left*, a high w-value leads to over-smoothing of the square and failure to detect the smaller ellipse to the upper right; *on the right*, a small w-value causes noisy artefacts.

4 A Level Set Method for Ductile Fracture[1]

4.1 Introduction

The scientific investigations of fracture mechanics have a long history, dating back to the study of metal fatigue during World War I by A. A. Griffith with his seminal work in [53]. Other well-known incidents are the frequent failures of Allied-built ships, see Figure 4.1, which were reduced by intensified fracture research. A more extensive history of fracture mechanics can be found in [37]. It is still a very active field; applications range from stress analysis of structural elements in power plants [117], airplanes [99] and bridge failures [35] to the development of advanced materials [78], and many more. Another application that has arisen fairly recently is the computer simulation of fracture propagation for graphics. It is used in movies, e.g. [57], as well as real-time applications and environments, like [111] and their demonstration in a Star Wars computer game.

Simulation of fracture and failure phenomena was introduced to computer graphics in the pioneering work of [142]. Early approaches typically made use of simple separation along mesh element boundaries [84, 96, 101, 129] or even element deletion [44]. The available geometric detail in this type of approach was increased somewhat by subdivision of elements in the mesh prior to splitting [15, 94]; however, those approaches had the tendency to introduce elements with poor aspect ratios. More geometrically rich fracture patterns were generated by allowing failure along more arbitrary paths (albeit with the expense of re-meshing) [98, 103, 105]. Embedded methods have been developed to minimize the complexity of re-meshing by embedding material surfaces into the existing mesh [6, 47, 93, 95, 128]. Although these works generalized the approach to fracture, the embedding idea goes back at least to free form deformations [26, 43, 125, 140]. Also, particle-based methods can provide flexibility for topology change [112]. Computer graphics approaches primarily use a principal stress failure criterion [72, 93, 95, 96, 103,

[1] Chapter based on Publication [56]: J. Hegemann, C. Jiang, C. Schroeder, and J.M. Teran. A Level Set Method for Ductile Fracture. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 2013. Accepted.

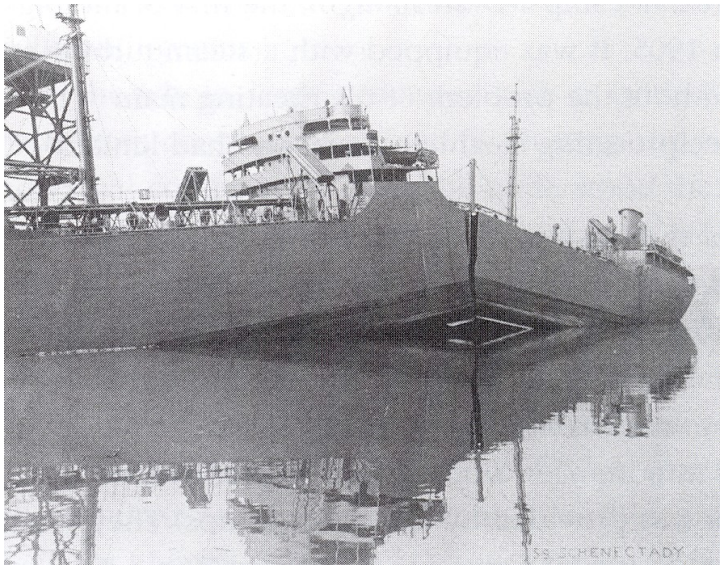


Figure 4.1: A T2 tanker split apart in harbor during World War II due to fracture (Source: `http://en.wikipedia.org/wiki/File:TankerSchenectady.jpg`).

105]. This has also been used for nearly rigid materials where an instantaneous linear elastic response after collision events was used to determine stresses [6, 135, 150]. Grain boundaries in this type of treatment can help to quickly create plausible fracture patterns [6, 57, 150]. Other interesting models for crack patterns were developed in [65, 66, 98].

Our focus is on ductile fracture of elasto-plastic solids. We use a level set method to evolve damaged regions of material with an embedded approach to reduce meshing complexity. Level set methods have proven very effective for handling topological changes for fluids and we show that they can also be used to reduce remeshing efforts for failure of solids. The level set evolves in material space to minimize Griffith's energy as an alternative to the principle stress criteria popular in computer graphics. This is a generalization of the work in [3] to large-strain, ductile materials. The level set description of the material region is used to simplify the determination of material connectivity in the embedded meshing approach from [128, 140] and is similar to the ideas used in [82]. We accurately compute the integrals in the FEM discretization of the elastic forces taking into account sub-cell geometric detail as is commonly done with discretizations using the extended finite element method (XFEM) [9, 119]. We provide a new mechanism for generating fragments of material in damaged regions (as defined by the level set evolution). Finally, we employ a material point method treatment of collision response.

In our approach, we extend the method of [3] from the quasistatic, linear elasticity to dynamics and arbitrary constitutive models. Second, we generalize this work to embedded geometries where the material boundary is initially defined from a level set. Also, we provide a new fragment generation algorithm to prevent volume loss inherent in [3]. This fragment generation procedure is specifically designed for an evolving level

set definition of healthy and damaged material. The resulting algorithm is significantly less complex than explicit remeshing strategies commonly used in computer graphics. Lastly, we demonstrate the application of the material point method (MPM) to the longstanding problem of embedded surface collisions. Collisions with these types of surfaces (e.g. resulting from marching tetrahedra) are notoriously difficult to resolve due to the inherently ill-conditioned sliver triangles arising from isosurface contouring. We also provide a significant improvement to the MPM approach with the addition of barycentrically bound ghost particles. These are used to improve material coverage of the background MPM grid in large deformation scenarios. In an additional investigation, we show that this modification can be done in a way to conserve total linear momentum.

4.2 Embedded Mesh and Duplication

As discussed in Section 2.3, we use a signed distance function ϕ in material coordinates $\boldsymbol{X}$ to create an embedded Lagrangian mesh for our material. The mesh consists of all tetrahedral elements in a regular background lattice with at least one node $\boldsymbol{X}_p$ having $\phi(\boldsymbol{X}_p) < 0$. However, since we want to allow for separation of different parts of the material, e.g. fragments broken off as part of the simulation, we will use connectivity in this mesh which is slightly different than that of the background lattice. We refer to any node incident on a boundary tetrahedron that has a positive ϕ value as a virtual node since it is outside but still participates in the discretization by virtue of the embedding. We define a boundary tetrahedron as one having nodes with both positive and negative ϕ values.

The level set will evolve to minimize the Griffith's energy of the material, which we will describe in Section 4.4. During this process, we treat the evolution as a phase change from "healthy" to "damaged" material. As this process occurs, we create a mesh for both damaged and healthy regions. To illustrate this, let ϕ define the healthy region at some time step prior to evolution (by way of describing the healthy material as the region where $\phi < 0$), and let $\hat{\phi}$ denote the new level set after evolution. Our energy evolution is defined so that material cannot transition from damaged to healthy. Therefore, the region with $\hat{\phi} < 0$ is contained in the region with $\phi < 0$ (and in fact $\hat{\phi} \geq \phi$). To create the healthy and damaged material meshes, we first create sub-element approximations to the zero isocontour of both ϕ and $\hat{\phi}$ using the previously described process (see Section 2.3) of triangle and quadrilateral insertion on boundary elements (as defined by the respective level sets). Note that both level sets will often cut the same tetrahedron, and therefore some elements may have material surfaces introduced by both level sets. Since our strict evolution from healthy to damaged enforces $\hat{\phi} \geq \phi$ (see Section 4.4), there will never be any crossings between these surfaces.

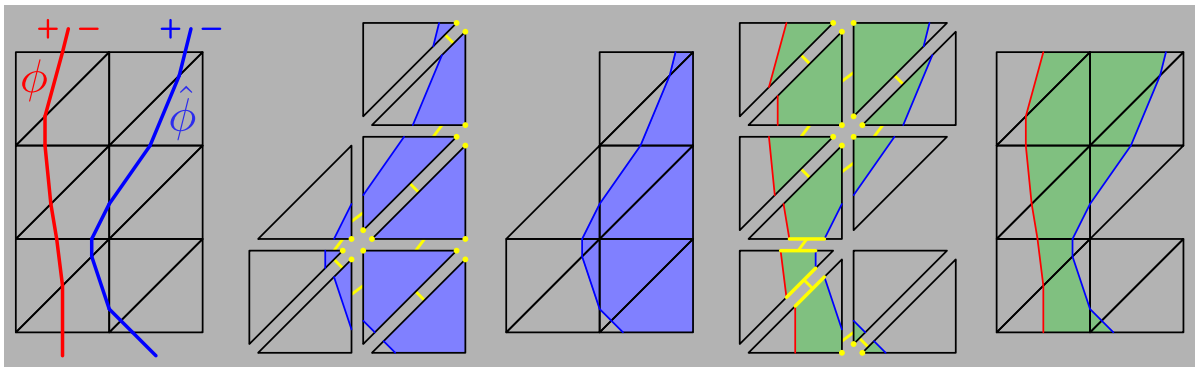


Figure 4.2: Elements are duplicated for the healthy (blue) and damaged (green) regions. The degrees of freedom along segments are merged if an endpoint is material on both segments (yellow dots) or if the segment is cut by both level sets for the damaged region (yellow lines).

We combine sub-element geometric information with the level set values to define the material connectivity of the computational mesh. Our procedure is similar to the connectivity criteria of [128, 140], but our procedure is specifically designed for an evolving level set definition of our healthy material and fragments generated in the evolution. First, we create a copy of each tetrahedron that has at least one node p with $\hat{\phi}(\boldsymbol{X}_p) < 0$. That is, this copy has its nodes disconnected from its neighbors with the introduction of potentially temporary virtual nodes. These elements will form the healthy mesh. Second, we create a copy for all tetrahedra that either have (i) a vertex with the product $\hat{\phi}(\boldsymbol{X}_p)\phi(\boldsymbol{X}_p) < 0$ (these nodes transitioned from healthy to damaged in the evolution from ϕ to $\hat{\phi}$) or (ii) an incident edge that is cut by both $\hat{\phi}$ and ϕ (these edges correspond to where the transition occurred at a material node, but without incurring a sign change). These tetrahedra comprise the newly damaged region as defined by the evolution. Note that some tetrahedra will give rise to copies for both damaged and healthy regions. This copying procedure is the same as in [128, 140]. In the final step, we merge nodes across element faces based on material connectivity. However, our means of establishing this is simplified greatly by our level set and sub-element geometric information. Specifically, faces of adjacent healthy tetrahedral elements (those copied based on the criterion that at least one node p has $\hat{\phi}(\boldsymbol{X}_p) < 0$) are merged if at least one original node on the face has $\hat{\phi} < 0$. Faces of adjacent damaged copies are merged if they either share a node that transitioned ($\hat{\phi}(\boldsymbol{X}_p)\phi(\boldsymbol{X}_p) < 0$) or if they share an edge that was cut twice. In cases where not all copies of a node are merged, multiple duplicates of a node remain and thus new degrees of freedom are added to the system. Similarly, the newly created fragments also give to new degrees of freedom. This duplication process allows the material to separate in accordance with material connectivity or lack thereof. The entire procedure of splitting and merging as well as fragment generation is illustrated in Figure 4.2.

During any time step, the portion of the domain undergoing fracture can be considered

as composed of two regions: a healthy region Ω_{h}^0 and a damaged region Ω_{d}^0 that is being shed from the healthy region through fracture. In addition, there are fragments created in previous time steps which are simulated but are not subject to further fracture. For simplicity, we will not mention these previously fractured regions further as they are simulated in a straightforward manner.

4.3 Elasto-Plastic Dynamics

For this method, we define our elasto-plastic constitutive relation from the corotational energy density in [88] (similar but different constitutive models can be found in [133]) as

$$\tilde{\Psi}(\boldsymbol{F}) = \mu\|\boldsymbol{F} - \boldsymbol{R}\|_{\mathrm{F}}^2 + \frac{\lambda}{2}(\mathrm{tr}(\boldsymbol{S} - \boldsymbol{I}))^2, \tag{4.1}$$

where $\boldsymbol{F} = \boldsymbol{R}\boldsymbol{S}$ is the polar decomposition of the deformation gradient and $\|\cdot\|_{\mathrm{F}}$ the Frobenius norm

$$\|\boldsymbol{A}\|_{\mathrm{F}} = \sqrt{\sum_i^m \sum_j^n |A_{ij}|^2} \qquad \text{for } \boldsymbol{A} \in \mathbb{R}^{m\times n}. \tag{4.2}$$

In practice, we compute the singular value decomposition (SVD) of $\boldsymbol{F}$

$$\boldsymbol{F} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}, \tag{4.3}$$

which allows us to obtain (see [51])

$$\boldsymbol{R} = \boldsymbol{U}\boldsymbol{V}^{\mathrm{T}}, \tag{4.4}$$

and with that

$$\boldsymbol{S} = \boldsymbol{R}^{\mathrm{T}}\boldsymbol{F}. \tag{4.5}$$

There are many algorithms to compute the SVD, see e.g. [51]. We use the one described in [86].

For this constitutive model, we can now specify derivatives of the energy density $\Psi = \tilde{\Psi}$ as needed according to Section 2.4. The first derivative is easily computed as

$$\boldsymbol{P} = 2\mu\,(\boldsymbol{F} - \boldsymbol{R}) + \lambda(\mathrm{tr}(\boldsymbol{R}^{\mathrm{T}}\boldsymbol{F} - \boldsymbol{I}))\boldsymbol{R}. \tag{4.6}$$

For the second derivative, we first formulate the energy density of (4.1) in terms of the

singular values σ_i:

$$\hat{\Psi}(\boldsymbol{\sigma}) = \mu \sum_i^d (\sigma_i - 1)^2 + \frac{\lambda}{2} \left(\sum_i^d (\sigma_i - 1) \right)^2 . \tag{4.7}$$

As mentioned before, we can differentiate $\hat{\Psi}$ very easily with respect to the σ_i, yielding

$$\frac{\partial \hat{\Psi}}{\partial \sigma_i} = 2\mu(\sigma_i - 1) + \lambda \sum_i^d (\sigma_i - 1) \tag{4.8}$$

$$\frac{\partial^2 \hat{\Psi}}{\partial \sigma_i \partial \sigma_j} = 2\mu\delta_{ij} + \lambda\sigma_i \tag{4.9}$$

and substitute these into the framework outlined in Section 2.4.4 to obtain $\frac{\partial \boldsymbol{P}}{\partial \boldsymbol{F}}(\boldsymbol{\Sigma})$, which we will once again label $\hat{G}$ for notational convenience. Most of its components are zero, except

$$\hat{G}_{1111} = 2\mu + \lambda, \tag{4.10}$$

$$\hat{G}_{2222} = 2\mu + \lambda, \tag{4.11}$$

$$\hat{G}_{3333} = 2\mu + \lambda, \tag{4.12}$$

$$\hat{G}_{1122} = \hat{G}_{2211} = \lambda, \tag{4.13}$$

$$\hat{G}_{1133} = \hat{G}_{3311} = \lambda, \tag{4.14}$$

$$\hat{G}_{2233} = \hat{G}_{3322} = \lambda, \tag{4.15}$$

$$\hat{G}_{1221} = \hat{G}_{2112} = \frac{3\lambda + 2\mu - \lambda\sigma_3}{\sigma_1 + \sigma_2} - \lambda, \tag{4.16}$$

$$\hat{G}_{1331} = \hat{G}_{3113} = \frac{3\lambda + 2\mu - \lambda\sigma_2}{\sigma_1 + \sigma_3} - \lambda, \tag{4.17}$$

$$\hat{G}_{2332} = \hat{G}_{3223} = \frac{3\lambda + 2\mu - \lambda\sigma_1}{\sigma_2 + \sigma_3} - \lambda, \tag{4.18}$$

$$\hat{G}_{1212} = \hat{G}_{2121} = 2\mu + \lambda - \frac{3\lambda + 2\mu - \lambda\sigma_3}{\sigma_1 + \sigma_2}, \tag{4.19}$$

$$\hat{G}_{1313} = \hat{G}_{3131} = 2\mu + \lambda - \frac{3\lambda + 2\mu - \lambda\sigma_2}{\sigma_1 + \sigma_3}, \tag{4.20}$$

$$\hat{G}_{2323} = \hat{G}_{3232} = 2\mu + \lambda - \frac{3\lambda + 2\mu - \lambda\sigma_1}{\sigma_2 + \sigma_3}, \tag{4.21}$$

As detailed in Section 2.4, this suffices to implement an implicit backward Euler scheme.

4.3.1 Plasticity

In addition to the discretization outlined in Section 2.4, we incorporate a plasticity model for some of our simulations. Plasticity describes an irreversible deformation (as opposed

to a fully elastic one). This behavior typically occurs after some initial reversible stage is exceeded. The plastic regime itself is often similarly limited to a certain range of deformation, depending on the material.

To accurately model this kind of behavior, we use the multiplicative decomposition of the deformation gradient into an elastic and a plastic part as discussed in [67]:

$$\boldsymbol{F} = \boldsymbol{F}_\mathrm{e}\boldsymbol{F}_\mathrm{p}. \tag{4.22}$$

For a given $\boldsymbol{F}$, we obtain $\boldsymbol{F}_\mathrm{e}$ via (4.22) and compute its SVD:

$$\boldsymbol{F}_\mathrm{e} = \boldsymbol{U}_\mathrm{e}\boldsymbol{\Sigma}_\mathrm{e}\boldsymbol{V}_\mathrm{e}^\mathrm{T} \tag{4.23}$$

We clamp the singular values to $[\sigma_{\mathrm{min,e}}, \sigma_{\mathrm{max,e}}]$. For the limiting values, we choose

$$\sigma_{\mathrm{min,e}} = \frac{1}{\sigma_\mathrm{e}}, \tag{4.24}$$

$$\sigma_{\mathrm{max,e}} = \sigma_\mathrm{e} \tag{4.25}$$

for a given value σ_e, which controls the limitation of the initial elastic phase of the deformation. We denote $\hat{\boldsymbol{\Sigma}}_\mathrm{e} = \mathrm{clamp}(\boldsymbol{\Sigma}_\mathrm{e})$ and compute

$$\hat{\boldsymbol{F}}_\mathrm{e} = \boldsymbol{U}_\mathrm{e}\hat{\boldsymbol{\Sigma}}_\mathrm{e}\boldsymbol{V}_\mathrm{e}^\mathrm{T} \tag{4.26}$$

as well as

$$\hat{\boldsymbol{F}}_\mathrm{p} = \boldsymbol{V}_\mathrm{e}\hat{\boldsymbol{\Sigma}}_\mathrm{e}^{-1}\boldsymbol{U}_\mathrm{e}^\mathrm{T}\boldsymbol{F}. \tag{4.27}$$

For the thus obtained $\hat{\boldsymbol{F}}_\mathrm{p}$, we once more compute the SVD:

$$\hat{\boldsymbol{F}}_\mathrm{p} = \boldsymbol{U}_\mathrm{p}\boldsymbol{\Sigma}_\mathrm{p}\boldsymbol{V}_\mathrm{p}^\mathrm{T}. \tag{4.28}$$

If the singular values in $\boldsymbol{\Sigma}_\mathrm{p}$ do not exceed another limit criterion, we update $\boldsymbol{F}_\mathrm{e} = \hat{\boldsymbol{F}}_\mathrm{e}$ and $\boldsymbol{F}_\mathrm{p} = \hat{\boldsymbol{F}}_\mathrm{p}$, and store

$$\boldsymbol{F}_\mathrm{p}^{-1} = \boldsymbol{V}_\mathrm{p}\boldsymbol{\Sigma}_\mathrm{p}^{-1}\boldsymbol{U}_\mathrm{p}^\mathrm{T} \tag{4.29}$$

for the next iteration.

In place of the energy density defined by (4.1), we reflect the change to the elastic response due to plasticity by defining the energy density

$$\Psi(\boldsymbol{F}) = \tilde{\Psi}(\boldsymbol{F}\boldsymbol{F}_\mathrm{p}^{-1}). \tag{4.30}$$

For notational convenience, we also write

$$\Psi_{\mathrm{e}}(\boldsymbol{F}_{\mathrm{e}}) = \tilde{\Psi}(\boldsymbol{F} = \boldsymbol{F}_{\mathrm{e}}). \tag{4.31}$$

Its derivative with respect to $\boldsymbol{F}$ is given by the chain rule as

$$\begin{aligned}
P_{ik} &= \frac{\partial \Psi}{\partial F_{ik}} \\
&= \sum_{rs} \frac{\partial \Psi_{\mathrm{e}}}{\partial (F_{\mathrm{e}})_{rs}} \frac{\partial (F_{\mathrm{e}})_{rs}}{\partial F_{ik}} \\
&= \sum_{rs} \frac{\partial \Psi_{\mathrm{e}}}{\partial (F_{\mathrm{e}})_{rs}} \sum_{t} \frac{\partial}{\partial F_{ik}} (F_{rt} (F_{\mathrm{p}}^{-1})_{ts}) \\
&= \sum_{rs} \frac{\partial \Psi_{\mathrm{e}}}{\partial (F_{\mathrm{e}})_{rs}} \sum_{t} (F_{\mathrm{p}}^{-1})_{ts} \delta_{ir} \delta_{kt} \\
&= \sum_{s} \frac{\partial \Psi_{\mathrm{e}}}{\partial (F_{\mathrm{e}})_{is}} (F_{\mathrm{p}}^{-1})_{ks} \\
\Rightarrow \boldsymbol{P} &= \boldsymbol{P}_{\mathrm{e}}(\boldsymbol{F}_{\mathrm{e}}) \boldsymbol{F}_{\mathrm{p}}^{-\mathrm{T}}.
\end{aligned} \tag{4.32}$$

Similarly, we can then compute the Hessian as

$$\begin{aligned}
\frac{\partial P_{ik}}{\partial F_{jm}} &= \frac{\partial}{\partial F_{jm}} \sum_{\ell} (P_{\mathrm{e}})_{i\ell} (F_{\mathrm{p}}^{-\mathrm{T}})_{\ell k} \\
&= \sum_{\ell} \sum_{rs} \frac{\partial (P_{\mathrm{e}})_{i\ell}}{\partial (F_{\mathrm{e}})_{rs}} \frac{\partial (F_{\mathrm{e}})_{rs}}{\partial F_{jm}} (F_{\mathrm{p}}^{-\mathrm{T}})_{\ell k} \\
&= \sum_{\ell} \sum_{rs} \frac{\partial (P_{\mathrm{e}})_{i\ell}}{(\partial F_{\mathrm{e}})_{rs}} (F_{\mathrm{p}}^{-1})_{ms} \delta_{rj} (F_{\mathrm{p}}^{-\mathrm{T}})_{\ell k} \\
\Rightarrow \frac{\partial P_{ik}}{\partial F_{jm}} &= \sum_{\ell,s} \frac{\partial (P_{\mathrm{e}})_{i\ell}}{\partial (F_{\mathrm{e}})_{js}} (F_{\mathrm{p}}^{-1})_{ms} (F_{\mathrm{p}}^{-1})_{k\ell}.
\end{aligned} \tag{4.33}$$

The values of $\boldsymbol{P}_{\mathrm{e}}(\boldsymbol{F}_{\mathrm{e}})$ and $\frac{\partial (P_{\mathrm{e}})_{i\ell}}{\partial (F_{\mathrm{e}})_{js}}(\boldsymbol{F}_{\mathrm{e}})$ are equivalent to the corresponding ones of the non-plastic case and thus we have all components readily available to evaluate (4.32) and (4.33).

4.4 Griffith's Energy Evolution

In his famous work [53], Griffith developed an energy based approach to understand formerly unexplained fracture phenomena. He incorporated an energy balance approach to the formation of new fracture surfaces. To illustrate the idea, consider a loaded elastic

plate with an initial crack of length a. Its total energy U can be written as

$$U = U_0 + U_a + U_\gamma - F$$

with

U_0: total elastic energy of the loaded but uncracked plate (constant),

U_a: change in the stored elastic energy due to the formation of the initial crack; introducing the crack causes the plate to lose stiffness and therefore potential energy, so $U_\mathrm{a} < 0$,

U_γ: change in the elastic surface energy due to the introduction of new surface along the crack; energy is consumed to disrupt the chemical bonds between the molecules of the material and stored in the new surface, so $U_\gamma > 0$,

F : work performed by external forces.

Crack growth can only occur if an increment of the crack length a decreases the stored energy, which yields

$$\frac{\mathrm{d}U}{\mathrm{d}a} \leq 0.$$

With the definition of U, and U_0 being constant this means

$$\frac{\mathrm{d}}{\mathrm{d}a}(U_a + U_\gamma - F) \leq 0,$$

which can be rearranged to

$$\frac{\mathrm{d}}{\mathrm{d}a}(F - U_a) \geq \frac{\mathrm{d}U_\gamma}{\mathrm{d}a},$$

where

$\frac{\mathrm{d}F}{\mathrm{d}a}$: energy provided by external work per crack increment $\mathrm{d}a$,

$\frac{\mathrm{d}U_a}{\mathrm{d}a}$: elastic energy released by a potential crack increment $\mathrm{d}a$.

It thus follows that

$\left(\frac{\mathrm{d}F}{\mathrm{d}a} - \frac{\mathrm{d}U_a}{\mathrm{d}a}\right)$: energy available for crack growth,

$\frac{\mathrm{d}U_\gamma}{\mathrm{d}a}$: surface energy of the crack surface per crack increment $\mathrm{d}a$.

In other words: In order for the crack to grow the energy provided by external forces and released from the plate by a crack increment must exceed the energy needed to form the new crack surfaces.

Griffith's idea was revisited by Francfort and Marigo in [46], and again in [45], to bypass the requirements of an initial crack as well as a well-defined crack path. Their model formulated quasistatic brittle fracture as a global energy minimization problem. Although such a global energy minimization postulate is not proven for real fracture propagation, it is common practice in contemporary material science and we operate under the same assumption.

Basing on these models, we use a Griffith-type energy minimization as our fracture evolution criterion similar to the method developed by Allaire et al in [3]. Reusing the notation for the healthy region $\Omega_h^0 = \{\boldsymbol{X} | \phi(\boldsymbol{X}) < 0\}$ and the damaged $\Omega_d^0 = \{\boldsymbol{X} | \phi(\boldsymbol{X}) > 0\}$, the Griffith's energy is defined as

$$E_{\mathrm{G}}(\phi) = \int_{\Omega_{\mathrm{h}}^0} \Psi \, d\boldsymbol{X} + \int_{\Omega_{\mathrm{d}}^0} \kappa \, d\boldsymbol{X}. \tag{4.34}$$

The coefficient κ is the rate of Griffith's energy release. In the damaged region, it represents the energy that was necessary to create the damage to this part of material. It can be used to limit the tendency to shrink Ω_{h}^0 and is therefore intuitively used to increase the material's resistance to fracture. Without this term, we could easily release the elastic energy stored in Ω_{h}^0 by evolving until we had $\Omega_{\mathrm{h}}^0 = \emptyset$.

In order to obtain the speed for a level set evolution, we interpret E_{G} as a functional in Ω_{h}^0 and use Lemma 2.2 from Section 2.1 to differentiate (4.34) with respect to Ω_{h}^0:

$$E'_{\mathrm{G}}(\Omega_{\mathrm{h}}^0)[\theta] = \int_{\partial\Omega_{\mathrm{h}}^0} \Psi(\boldsymbol{X}, \boldsymbol{F}(\boldsymbol{X}))\theta \cdot \boldsymbol{n}^0(\boldsymbol{X}) \, \mathrm{d}s(\boldsymbol{X}) + \int_{\partial(\Omega \setminus \Omega_{\mathrm{h}}^0)} \kappa\theta \cdot \boldsymbol{n}^1(\boldsymbol{X}) \, \mathrm{d}s(\boldsymbol{X}). \tag{4.35}$$

Since $\partial(\Omega \setminus \Omega_{\mathrm{h}}^0) = \partial\Omega_{\mathrm{h}}^0$ and $\boldsymbol{n}^0 = -\boldsymbol{n}^1$ where $\theta \neq \boldsymbol{0}$, we can rewrite this as

$$E'_{\mathrm{G}}(\Omega_{\mathrm{h}}^0)[\theta] = \int_{\partial\Omega_{\mathrm{h}}^0} (\Psi(\boldsymbol{X}, \boldsymbol{F}(\boldsymbol{X})) - \kappa)\theta \cdot \boldsymbol{n}^0(\boldsymbol{X}) \, \mathrm{d}s(\boldsymbol{X}) \tag{4.36}$$

As discussed in Section 2.2, the level set function ϕ representing the boundary of Ω_{h}^0 can be evolved through a pseudo-time by solving

$$\frac{\partial \phi}{\partial t} - \ \delta_\epsilon(\phi) V = 0, \tag{4.37}$$

and the speed can now be chosen as

$$V(\boldsymbol{X}) = \Psi(\boldsymbol{X}, \boldsymbol{F}(\boldsymbol{X})) - \kappa. \tag{4.38}$$

Finally, to ensure that material only transitions from healthy to damaged, we disregard any change in ϕ at nodes where its value decreases as this would correspond to a growth in the healthy region and thus violate the irreversibility of fracture, i.e. we enforce

$$V(\boldsymbol{X}) = \max(0, \Psi(\boldsymbol{X}, \boldsymbol{F}(\boldsymbol{X})) - \kappa). \tag{4.39}$$

The obtained velocity corresponds again to the idea of Griffith that a transition from healthy phase to damaged phase only occurs if the release of elastic energy exceeds a threshold κ. An illustration of the energy evolution is given in Figure 4.4 among the results in Section 4.7.

Since the level set function is defined on the mesh nodes, we need to compute the energy density Ψ on the nodes as well. We do this by computing its value as specified in Section 4.3 within each mesh simplex S^α (where it is piecewise constant), and then employing a weighted average over all elements that contain this node:

$$\Psi(\mathbf{X}_p) = \frac{\sum_{\alpha | p \in S^\alpha} \Psi(S^\alpha) \int_{S^\alpha} N_p(\mathbf{X}) \, \mathrm{d}\mathbf{X}}{\sum_{\alpha | p \in S^\alpha} \int_{S^\alpha} N_p(\mathbf{X}) \, \mathrm{d}\mathbf{X}}, \tag{4.40}$$

with N_p being the linear basis function of node p.

In practice, we enforce a CFL restriction on Δs so that ϕ does not move the boundary of the healthy region by more than one mesh spacing in one pseudo-time step. This results in the necessity for multiple executions of the update (2.7), though sometimes it might be more visually pleasing to use only a limited number of steps.

Furthermore, to avoid artifacts and since the energy values driving the level set update are often very large, we not only reinitialize ϕ after every update but also take extra precautions while doing so. Before we apply the fast sweeping method detailed in Section 2.2.1, we first identify the location of the new interface, as defined by the new $\hat{\phi}$, solution to the level set update (4.37), within the boundary elements of the mesh as detailed in Section 2.3. We then use the embedded surface triangles to recompute the exact distance from the surface to the mesh vertices of the containing boundary elements. This re-evaluation procedure is only necessary at nodes where the level set value, and thus the interface, has changed, i.e. if $|\phi(\boldsymbol{X}_i) - \hat{\phi}(\boldsymbol{X}_i)| > \epsilon$, where ϵ can be chosen as a multiple of machine precision. We then use these exact distances around the interface in the initialization phase of the more efficient fast sweeping method to propagate the signed distance property to all other nodes of the mesh.

4.5 Collisions

Collisions between different (parts of) bodies are a problem completely separate from the PDE governing the elasticity. The method of Bridson et al in [20] has become common use in the computer graphics community; however, it poses fairly restrictive requirements on the surface triangles of the colliding interfaces. Since the fracture surfaces can become of arbitrary quality throughout their evolution, those required properties cannot be guaranteed throughout our simulations. We therefore need a more robust approach. The material point method (MPM) (e.g. [32, 136]) has been successfully used for collision handling in fracture settings. However, we do not use an MPM discretization but rather borrow parts of the ideas of Huang et al in [64] and use them to resolve collisions after we advanced positions and velocities via an implicit FEM update.

We use an impulse-based response for rigid collision bodies and self-collision. This is difficult because the embedded meshes that define the material regions have many sliver triangles since they are generated by marching tetrahedra. We found that the material point method for collision impulses outlined in [64] was effective at applying impulses when the surface geometry contained sliver elements. This technique uses the gradients of interpolating functions on a background regular Cartesian grid to estimate normals to the boundary of the material region. This is convenient because it does not rely on high-quality boundary geometry. However, this robustness does come at the expense of accuracy, and we alleviate this by augmenting the original method by seeding barycentrically bound ghost particles.

We will now outline this collision algorithm in more detail. For simplicity, we will restrict our description to two colliding bodies, but any number of objects is possible. We compute the connected components of all mesh particles based on material connectivity as given by the mesh. We will use a subscript b to indicate that we store the contributions to a grid node i separately for each of the two bodies.

4.5.1 Ghost Particles

As we will see in Section 4.5.2, the accuracy of the collision algorithm depends on the number of material particles that contribute to any affected grid node. That is why we increase the number of material particles by creating *ghost particles* in every mesh element in addition to the degrees of freedom of the mesh. These dependent particles only exist for the purpose of collisions and do not contribute to the elasticity simulation. We use R and G to denote real and ghost particles, with $P = R \cup G$ (or R_b, G_b, and P_b when referring only to those particles in body b). The ghost particles do not represent any new degrees of freedom but are solely defined by their barycentric relation to their parent mesh vertices, with the barycentric weight $w_r^q = 0$ if ghost particle $q \in G$ is not

bound to real particle $r \in R$. We will further use the short-hand notation $q \in S^\alpha$ if q is bound to the vertices of S^α. The number of ghost particles per element S^α as well as their barycentric weights w_r^q can be determined a priori. We found in our numerical experiments that nine ghost particles per element with the following weights present a nice balance between improved accuracy and computational effort:

$$(w_r^1)_r = (0.25,\, 0.25,\, 0.25,\, 0.25), \tag{4.41}$$
$$(w_r^2)_r = (0.1,\, 0.3,\, 0.3,\, 0.3), \tag{4.42}$$
$$(w_r^3)_r = (0.3,\, 0.1,\, 0.3,\, 0.3), \tag{4.43}$$
$$(w_r^4)_r = (0.3,\, 0.3,\, 0.1,\, 0.3), \tag{4.44}$$
$$(w_r^5)_r = (0.3,\, 0.3,\, 0.3,\, 0.1), \tag{4.45}$$
$$(w_r^6)_r = (0.4,\, 0.2,\, 0.2,\, 0.2), \tag{4.46}$$
$$(w_r^7)_r = (0.2,\, 0.4,\, 0.2,\, 0.2), \tag{4.47}$$
$$(w_r^8)_r = (0.2,\, 0.2,\, 0.4,\, 0.2), \tag{4.48}$$
$$(w_r^9)_r = (0.2,\, 0.2,\, 0.2,\, 0.4). \tag{4.49}$$

The positions and candidate velocities of ghost particles are computed in a straightforward manner as the linear combination of their parents, i.e.

$$\boldsymbol{x}_q^n = \sum_{r \in R} w_r^q \boldsymbol{x}_r^n, \tag{4.50}$$
$$\bar{\boldsymbol{v}}_q^{n+1} = \sum_{r \in R} w_r^q \bar{\boldsymbol{v}}_r^{n+1}, \tag{4.51}$$

where $q \in G$. With $\bar{\boldsymbol{v}}_r^{n+1}$ we denote the candidate velocities at time t^{n+1}, solution to the elasticity update of Section 4.3. These velocities might change during the collision processing.

The computation of the mass for ghost particles has to be done slightly more carefully to conserve total mass. The particles masses $m_{r\alpha}^n$ corresponding to an element S^α and a real particle r are computed respecting the embedded boundaries as the integral

$$m_{r\alpha}^n = \int_{S^\alpha} \rho_0 N_r \, \mathrm{d}\boldsymbol{X}. \tag{4.52}$$

The mass associated with a node r can thus change if material gets damaged or breaks off during the fracture evolution. The mass $\sum_k m_{r\alpha}^n$ of real particle r is distributed to r and embedded particles q proportional to their barycentric weights w_r^q (where $w_r^r = 1$)

so that

$$m_q^n = \sum_{r \in R} \frac{w_r^q m_{r\alpha}^n}{W_{r\alpha}} \tag{4.53}$$

$$m_r^n = \sum_{\alpha} \frac{m_{r\alpha}^n}{W_{r\alpha}} \tag{4.54}$$

with

$$W_{r\alpha} = 1 + \sum_{q \in G \cap S^\alpha} w_r^q, \tag{4.55}$$

where $r \in R$, $q \in G \cap S^\alpha$. The sum of all contributions $W_{r\alpha}$ is chosen so that

$$\begin{aligned}
\sum_{p \in P} m_p^n &= \sum_{r \in R} m_r^n + \sum_{q \in G} m_q^n \\
&= \sum_{r \in R} m_r^n + \sum_{\alpha} \sum_{q \in G \cap S^\alpha} m_q^n \\
&= \sum_{r \in R} \sum_{\alpha} \frac{m_{r\alpha}^n}{W_{r\alpha}} + \sum_{\alpha} \sum_{q \in G \cap S^\alpha} \sum_{r \in R} \frac{w_r^q m_{r\alpha}^n}{W_{r\alpha}} \\
&= \sum_{r \in R} \sum_{\alpha} \frac{m_{r\alpha}^n}{W_{r\alpha}} \underbrace{\left(1 + \sum_{q \in G \cap S^\alpha} w_r^q\right)}_{=W_{r\alpha}} \\
&= \sum_{r \in R} \sum_{\alpha} m_{r\alpha}^n \\
&= \sum_{r \in R} \sum_{\alpha} \int_{S^\alpha} \rho_0 N_r(\boldsymbol{X}) \, \mathrm{d}\boldsymbol{X} \\
&= \sum_{\alpha} \int_{S^\alpha} \rho_0 \, \mathrm{d}\boldsymbol{X}
\end{aligned}$$

accounts for the total mass.

4.5.2 Rasterization

Next, we rasterize the particle masses to the collision grid as

$$m_{bi}^n = \sum_{p \in P_b} m_p^n S_i(\boldsymbol{x}_p^n), \tag{4.56}$$

where the S_i are the standard continuous and piecewise trilinear (grid-based) nodal shape functions with the property

$$S_i(\boldsymbol{x}_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{else} \end{cases} \tag{4.57}$$

for grid nodes $\boldsymbol{x}_j$. We use the (deformed) positions $\boldsymbol{x}_p^n$ of the previous time step, t^n, as the values of the current time step will depend on any changes we make to the velocity field to avoid collisions.

The nodal velocity on the grid is computed as the ratio of rasterized momentum to mass

$$\bar{\boldsymbol{v}}_{bi}^{n+1} = \frac{\sum_{p \in P_b} m_p^n \bar{\boldsymbol{v}}_p^{n+1} S_i(\boldsymbol{x}_p^n)}{m_{bi}^n}. \tag{4.58}$$

We use the gradient of the nodal mass to find the outward normals of grid node i for body b

$$\bar{\boldsymbol{n}}_{bi}^n = \frac{\nabla m_{bi}^n}{\|\nabla m_{bi}^n\|} = \frac{\sum_{p \in P_b} m_p^n \nabla S_i(\boldsymbol{x}_p^n)}{\| \sum_{p \in P_b} m_p^n \nabla S_i(\boldsymbol{x}_p^n)\|}. \tag{4.59}$$

Note that for some grid nodes, the material barely overlaps with the support of the associated shape function, and ghost particles cannot offer any improvements. These nodes also barely contribute to the velocity field due to their low mass weights, so in practice this ghost particle strategy leads to acceptable results.

4.5.3 Contact Handling on the Grid

If particles from the two different bodies register at the same grid node i, a collision may occur. All contact velocities are relative to the center of mass at this grid node

$$\bar{\boldsymbol{v}}_i^{\text{com},n+1} = \frac{m_{1,i}^n \bar{\boldsymbol{v}}_{1,i}^{n+1} + m_{2,i}^n \bar{\boldsymbol{v}}_{2,i}^{n+1}}{m_{1,i}^n + m_{2,i}^n}. \tag{4.60}$$

The normals of the two bodies as defined by (4.59), $\boldsymbol{n}_{1,i}^n$ and $\boldsymbol{n}_{2,i}^n$, might not be colinear. However, this is necessary to guarantee conservation of momentum (see below). Therefore, we use the average of the two normals

$$\boldsymbol{n}_{1,i}^n = \frac{\bar{\boldsymbol{n}}_{1,i}^n - \bar{\boldsymbol{n}}_{2,i}^n}{\|\bar{\boldsymbol{n}}_{1,i}^n - \bar{\boldsymbol{n}}_{2,i}^n\|} \tag{4.61}$$

and set

$$\boldsymbol{n}_{2,i}^n = -\boldsymbol{n}_{1,i}^n. \tag{4.62}$$

(Other choices are possible, e.g. in case of significant differences in volume or stiffness, see [64]. We employ a different choice when colliding with a rigid body, see Subsection 4.5.5.) With these normals, we can determine approach and departure of the two bodies. An approach happens and collision may occur if

$$(\bar{\boldsymbol{v}}_{bi}^{n+1} - \boldsymbol{v}_i^{\mathrm{com},n+1}) \cdot \boldsymbol{n}_{bi}^n > 0. \tag{4.63}$$

This condition is equivalent for $b = 1$ and $b = 2$:

$$\begin{aligned}
& (\bar{\boldsymbol{v}}_{1,i}^{n+1} - \boldsymbol{v}_i^{\mathrm{com},n+1}) \cdot \boldsymbol{n}_{1,i}^n > 0 \\
\Leftrightarrow \quad & (\bar{\boldsymbol{v}}_{1,i}^{n+1} - \frac{m_{1,i}\bar{\boldsymbol{v}}_{1,i}^{n+1} + m_{2,i}\bar{\boldsymbol{v}}_{2,i}^{n+1}}{m_{1,i} + m_{2,i}}) \cdot \boldsymbol{n}_{1,i}^n > 0 \\
\Leftrightarrow \quad & (\frac{m_{1,i}\bar{\boldsymbol{v}}_{1,i}^{n+1} + m_{2,i}\bar{\boldsymbol{v}}_{1,i}^{n+1} - (m_{1,i}\bar{\boldsymbol{v}}_{1,i}^{n+1} + m_{2,i}\bar{\boldsymbol{v}}_{2,i}^{n+1})}{m_{1,i} + m_{2,i}}) \cdot \boldsymbol{n}_{1,i}^n > 0 \\
\Leftrightarrow \quad & \frac{m_{2,i}}{m_{1,i} + m_{2,i}}(\bar{\boldsymbol{v}}_{1,i}^{n+1} - \bar{\boldsymbol{v}}_{2,i}^{n+1}) \cdot \boldsymbol{n}_{1,i}^n > 0.
\end{aligned}$$

Since both bodies were rasterized to node i, we have $\frac{m_{2,i}}{m_{1,i}+m_{2,i}} > 0$ and $\frac{m_{1,i}}{m_{1,i}+m_{2,i}} > 0$, thus

$$\Leftrightarrow \quad \frac{m_{1,i}}{m_{1,i} + m_{2,i}}(\bar{\boldsymbol{v}}_{1,i}^{n+1} - \bar{\boldsymbol{v}}_{2,i}^{n+1}) \cdot \boldsymbol{n}_{1,i}^n > 0$$

By construction, $\boldsymbol{n}_{1,i}^n = -\boldsymbol{n}_{2,i}^n$, yielding

$$\begin{aligned}
\Leftrightarrow \quad & \frac{m_{1,i}}{m_{1,i} + m_{2,i}}(\bar{\boldsymbol{v}}_{2,i}^{n+1} - \bar{\boldsymbol{v}}_{1,i}^{n+1}) \cdot \boldsymbol{n}_{2,i}^n > 0 \\
\Leftrightarrow \quad & (\frac{m_{1,i}\bar{\boldsymbol{v}}_{2,i}^{n+1} + m_{2,i}\bar{\boldsymbol{v}}_{2,i}^{n+1} - (m_{1,i}\bar{\boldsymbol{v}}_{1,i}^{n+1} + m_{2,i}\bar{\boldsymbol{v}}_{2,i}^{n+1})}{m_{1,i} + m_{2,i}}) \cdot \boldsymbol{n}_{2,i}^n > 0 \\
\Leftrightarrow \quad & (\bar{\boldsymbol{v}}_{2,i}^{n+1} - \boldsymbol{v}_i^{\mathrm{com},n+1}) \cdot \boldsymbol{n}_{2,i}^n > 0.
\end{aligned}$$

If contact according to (4.63) occurs, the candidate velocities $\bar{\boldsymbol{v}}_{bi}^{n+1}$ need to be corrected by applying an appropriate impulse

$$\boldsymbol{v}_{bi}^{n+1} = \bar{\boldsymbol{v}}_{bi}^{n+1} + \frac{\Delta t}{m_{bi}} \boldsymbol{f}_{bi}^{\mathrm{ct}}. \tag{4.64}$$

This corrected velocity $\boldsymbol{v}_{bi}^{n+1}$ must satisfy the impenetrability condition

$$\sum_b \boldsymbol{v}_{bi}^{n+1} \cdot \boldsymbol{n}_{bi}^n = 0. \tag{4.65}$$

We find the normal component of the collision force $\boldsymbol{f}_{bi}^{\mathrm{ct}}$ by substituting (4.65) into (4.64), which yields (using also (4.60))

$$\begin{aligned} f_i^{\mathrm{ct,n}} &= \boldsymbol{f}_{bi}^{\mathrm{ct}} \cdot \boldsymbol{n}_{bi}^n \\ &= \frac{1}{\Delta t} \frac{m_{1,i}^n m_{2,i}^n}{m_{1,i}^n + m_{2,i}^n} (\bar{\boldsymbol{v}}_{2,i}^{n+1} - \bar{\boldsymbol{v}}_{1,i}^{n+1}) \cdot \boldsymbol{n}_{1,i}^n \\ &= \frac{m_{bi}^n}{\Delta t} (\boldsymbol{v}_i^{\mathrm{com},n+1} - \bar{\boldsymbol{v}}_{bi}^{n+1}) \cdot \boldsymbol{n}_{bi}^n. \end{aligned} \tag{4.66}$$

For friction-less contact with no tangential component - as we assume it for object-object collisions - we can directly insert this result back into (4.64) to obtain our final corrected velocity:

$$\boldsymbol{v}_{bi}^{n+1} = \bar{\boldsymbol{v}}_{bi}^{n+1} - [(\bar{\boldsymbol{v}}_{bi}^{n+1} - \boldsymbol{v}_i^{\mathrm{com},n+1}) \cdot \boldsymbol{n}_{bi}^n] \boldsymbol{n}_{bi}^n. \tag{4.67}$$

A tangential component can be computed in a similar fashion, starting from a no-slip condition; the resulting force can then be modeled with Coulomb friction.

4.5.4 Interpolation back to the Particles

After all grid velocities are corrected as necessary via (4.67), we use the grid shape functions defined in (4.57) to compute the new velocities of the Lagrangian particles. We loop through all real particles $r \in R$, find the cube it is contained in and find its new value as

$$\boldsymbol{v}_r^{n+1} = \xi [\boldsymbol{v}_r^n + \sum_i S_i(\boldsymbol{x}_r^n)(\boldsymbol{v}_{bi}^{n+1} - \boldsymbol{v}_{bi}^n)] + (1 - \xi) \sum_i S_i(\boldsymbol{x}_r^n) \boldsymbol{v}_{bi}^{n+1} \tag{4.68}$$

with b such that $r \in b$; $\xi \in [0, 1]$ is a control parameter that defines the ratio between PIC (Particle-In-Cell method [42]) versus FLIP (Fluid-Implicit-Particle method [19]). For our simulations, we found that full FLIP, i.e. $\xi = 1$, serves our purposes best. Since typically only a small portion of the Lagrangian particles are involved in collisions, we do not need the numerical viscosity introduced by PIC for stability. On the other hand, FLIP preserves more of the dynamics (cf. [151]), which is desirable for our application.

The final displacements of the particles for this time step are then given by

$$\boldsymbol{u}_r^{n+1} = \boldsymbol{u}_r^n + \Delta t \boldsymbol{v}_r^{n+1}, \tag{4.69}$$

which is consistent with the backward Euler time discritization used in Section 2.4. These displacements and velocities are the ones used in our level set evolution as well as for the initial state in the next time step of the PDE solver. Once again, ghost particles were created only for the purpose of improved initialization; no velocity is interpolated back to them.

4.5.5 Ground and other Rigid Bodies

We use the above method not only for object-object collisions but also to model collisions with the ground and other rigid bodies. In those cases, only minor adjustments are made. First, instead of employing the average in (4.61) we use the (possibly analytic) normal defined by the rigid body, e.g. in case of ground collisions

$$\boldsymbol{n}_{bi}^n = -\boldsymbol{e}_z, \tag{4.70}$$

or in case of a ball with midpoint $\boldsymbol{x}_{\text{ball}}$

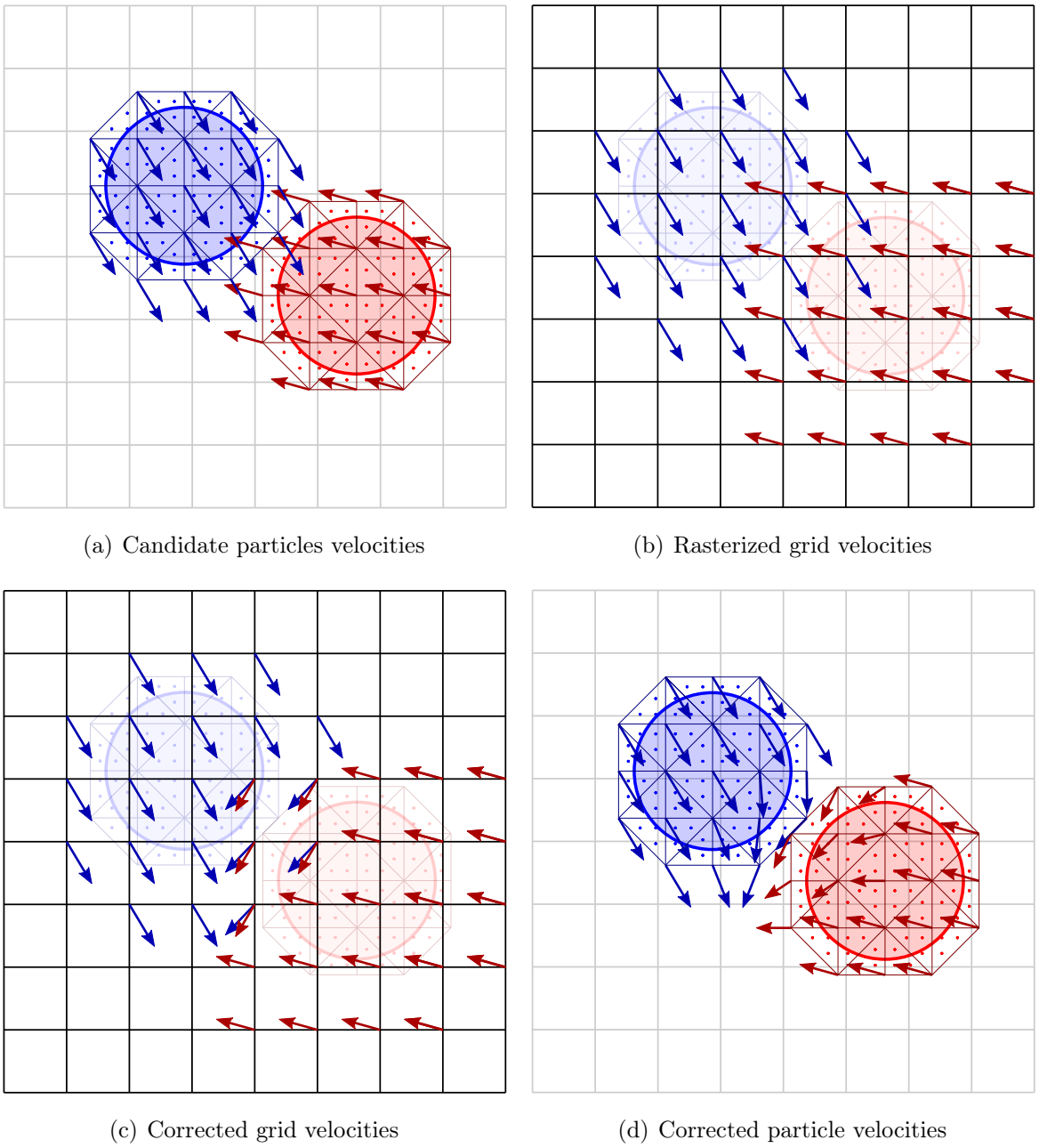
$$\boldsymbol{n}_{bi}^n = -\frac{\boldsymbol{x}_{bi}^n - \boldsymbol{x}_{\text{ball}}}{\|\boldsymbol{x}_{bi}^n - \boldsymbol{x}_{\text{ball}}\|}, \tag{4.71}$$

as outward normal for deformable body b. Further, we do not compute the center of mass velocity according to (4.60), but rather set it equal to that of the rigid body. For the special case of ground collisions, we can also introduce some friction by simply damping the x and y velocity component by a factor $\zeta \in [0, 1]$.

4.5.6 Collision Implementation Remarks

An illustration of the different steps of the collision handling is shown in Figure 4.3. We also point out the following remarks regarding its implementation:

- The background grid and all grid-based variables do not need to be stored, leading to no additional memory costs once collisions are resolved.
- The size of the background grid can easily be determined as the bounding box of all particle positions in deformed configuration.
- We choose the resolution of the background grid dependent on the resolution of the material mesh to maintain volume coverage, a typical value is $h_{\text{grid}} = 2h_{\text{mesh}}$.
- Since particles might be widely spread across the grid and not all grid nodes might be used, sparse data structures can be used for efficiency.



(a) Candidate particles velocities

(b) Rasterized grid velocities

(c) Corrected grid velocities

(d) Corrected particle velocities

Figure 4.3: From *top left* to *bottom right*: First, we rasterize the velocities from the Lagrangian particles (including additional ghost particles) to the Eulerian background grid; then, we detect where particles from different bodies register on the same grid point and project out their normal components; the corrected grid velocities are interpolated back to the mesh DoFs.

- The collision detection based on separate bodies does not cover collisions of different parts of the same piece of material. However, this limitation could be circumvented by subdividing a body into smaller regions that register separately onto the Eulerian grid. These subregions could then in turn collide with each other (neighboring ones excluded to not interfere with elasticity forces). However, for

our practical examples we found such an extension not necessary.

4.5.7 Conservation of Momentum and Suggested Improvements

In this subsection we will analyze the property of momentum conservation of our collision handling. This will lead to the observation of some limitations of the proposed addition of ghost particles. However, we will in turn develop some possible improvements that will eliminate the shortcomings. These findings are an extension to our publication [56] and go beyond its scope.

Lemma 4.1. *For the case $P = R$, i.e. without the use of any ghost particles, the grid based collision handling conserves linear momentum in each of its steps*

(i) rasterization,

(ii) collision response,

(iii) interpolation back to the mesh particles via FLIP.

Proof. (i) We will start with the rasterization process. Let

$$\boldsymbol{p}_{\text{mesh}} = \sum_{p \in P} m_p^n \bar{\boldsymbol{v}}_p^{n+1}$$

be the total linear momentum of the mesh. Then, we need to show that the mesh momentum is equal to the rasterized total grid momentum

$$\begin{aligned}
\boldsymbol{p}_{\text{grid}} &= \sum_{b,i} m_{bi}^n \bar{\boldsymbol{v}}_{bi}^{n+1} \\
&= \sum_{b,i} m_{bi}^n \frac{\sum_{p \in b} m_p^n \bar{\boldsymbol{v}}_p^{n+1} S_i(\boldsymbol{x}_p^n)}{m_{bi}^n} \\
&= \sum_{b,i} \sum_{p \in b} m_p^n \bar{\boldsymbol{v}}_p^{n+1} S_i(\boldsymbol{x}_p^n);
\end{aligned}$$

the summation can be reordered to yield

$$\begin{aligned}
\boldsymbol{p}_{\text{grid}} &= \sum_{p \in P} m_p^n \bar{\boldsymbol{v}}_p^{n+1} \underbrace{\sum_i S_i(\boldsymbol{x}_p^n)}_{= 1 \text{ due to the nature of the } S_i} \\
&= \boldsymbol{p}_{\text{mesh}}.
\end{aligned}$$

(ii) The collision handling as given by (4.67) conserves momentum at each node, i.e.

$$\sum_{b=1}^{2} m_{bi}^{n} \boldsymbol{v}_{bi}^{n+1} - \sum_{b=1}^{2} m_{bi}^{n} \bar{\boldsymbol{v}}_{bi}^{n+1} = 0. \tag{4.72}$$

To show that this equality holds, we substitute (4.67) into the left-hand side of (4.72), leading to

$$\begin{aligned}
&\sum_{b=1}^{2} m_{bi}^{n} \boldsymbol{v}_{bi}^{n+1} - \sum_{b=1}^{2} m_{bi}^{n} \bar{\boldsymbol{v}}_{bi}^{n+1} \\
&= \sum_{b=1}^{2} m_{bi}^{n} \left\{ \bar{\boldsymbol{v}}_{bi}^{n+1} - \left[(\bar{\boldsymbol{v}}_{bi}^{n+1} - \boldsymbol{v}_{i}^{\mathrm{com},n+1}) \cdot \boldsymbol{n}_{bi}^{n} \right] \boldsymbol{n}_{bi}^{n} \right\} - \sum_{b=1}^{2} m_{bi}^{n} \bar{\boldsymbol{v}}_{bi}^{n+1},
\end{aligned} \tag{4.73}$$

where the first two terms in each sum cancel each other out. Using $\boldsymbol{n}_{1,i}^{k} = -\boldsymbol{n}_{2,i}^{k}$ and the definition of $\boldsymbol{v}_{i}^{\mathrm{com},n+1}$ in (4.60) leads to

$$\begin{aligned}
&\sum_{b=1}^{2} m_{bi}^{n} \boldsymbol{v}_{bi}^{n+1} - \sum_{b=1}^{2} m_{bi}^{n} \bar{\boldsymbol{v}}_{bi}^{n+1} \\
&= \sum_{b=1}^{2} m_{bi}^{n} \left[(\boldsymbol{v}_{i}^{\mathrm{com},n+1} - \bar{\boldsymbol{v}}_{bi}^{n+1}) \cdot \boldsymbol{n}_{bi}^{n} \right] \boldsymbol{n}_{bi}^{n} && (4.74) \\
&= \left[\left(\sum_{b=1}^{2} m_{bi}^{n} (\boldsymbol{v}_{i}^{\mathrm{com},n+1} - \bar{\boldsymbol{v}}_{bi}^{n+1}) \right) \cdot \boldsymbol{n}_{1i}^{n} \right] \boldsymbol{n}_{1i}^{n} && (4.75) \\
&= \left[\left(\boldsymbol{v}_{i}^{\mathrm{com},n+1} \sum_{b=1}^{2} m_{bi}^{n} - \sum_{b=1}^{2} m_{bi}^{n} \bar{\boldsymbol{v}}_{bi}^{n+1} \right) \cdot \boldsymbol{n}_{1i}^{n} \right] \boldsymbol{n}_{1i}^{n} && (4.76) \\
&= \left[\left(\frac{\sum_{b=1}^{2} m_{bi}^{n} \bar{\boldsymbol{v}}_{bi}^{n+1}}{\sum_{b=1}^{2} m_{bi}^{n}} \sum_{b=1}^{2} m_{bi}^{n} - \sum_{b=1}^{2} m_{bi}^{n} \bar{\boldsymbol{v}}_{bi}^{n+1} \right) \cdot \boldsymbol{n}_{1i}^{n} \right] \boldsymbol{n}_{1i}^{n} && (4.77) \\
&= 0, && (4.78)
\end{aligned}$$

which concludes the statement of (4.72).

(iii) Lastly, the interpolation of the corrected velocities back to the particles via FLIP yields

$$\boldsymbol{p}_{\text{mesh}}^{\text{FLIP}} = \sum_{p\in P} m_p^n \boldsymbol{v}_p^{n+1} \tag{4.79}$$

$$= \sum_{p\in P} m_p^n \left(\bar{\boldsymbol{v}}_p^{n+1} + \sum_{b,p\in b} \sum_i S_i(\boldsymbol{x}_p^n)(\boldsymbol{v}_{bi}^{n+1} - \bar{\boldsymbol{v}}_{bi}^{n+1}) \right) \tag{4.80}$$

$$= \sum_{p\in P} m_p^n \bar{\boldsymbol{v}}_p^{n+1} + m_p^n \sum_{b,p\in b} \sum_i S_i(\boldsymbol{x}_p^n)(\boldsymbol{v}_{bi}^{n+1} - \bar{\boldsymbol{v}}_{bi}^{n+1}) \tag{4.81}$$

$$= \sum_{p\in P} m_p^n \bar{\boldsymbol{v}}_p^{n+1} + \underbrace{\sum_{b,i} (\boldsymbol{v}_{bi}^{n+1} - \bar{\boldsymbol{v}}_{bi}^{n+1}) \underbrace{\sum_{p\in b} m_p^n S_i(\boldsymbol{x}_p^n)}_{=m_{bi}^n}}_{=0 \text{ as shown above}} \tag{4.82}$$

$$= \boldsymbol{p}_{\text{mesh}}. \tag{4.83}$$

□

We will now see how ghost particles affect these momentum conservation properties. First, let us look at the momentum of the mesh with ghost particles

$$\boldsymbol{p}_{\text{mesh, ghost}} = \sum_{r\in R} m_r^n \bar{\boldsymbol{v}}_r^{n+1} + \sum_{q\in G} m_q^n \bar{\boldsymbol{v}}_q^{n+1} \tag{4.84}$$

$$= \sum_{r\in R} \left(\sum_\alpha \frac{m_{r\alpha}^n}{W_{r\alpha}} \right) \bar{\boldsymbol{v}}_r^{n+1} + \sum_\alpha \sum_{q\in S^\alpha} \left[\left(\sum_{r\in R} \frac{w_r^q m_{r\alpha}^n}{W_{r\alpha}} \right) \left(\sum_{r\in R} w_r^q \boldsymbol{v}_r^{n+1} \right) \right] \tag{4.85}$$

Here it becomes obvious that, unfortunately, the product of sums in the second term prevents $\boldsymbol{p}_{\text{mesh, ghost}}$ from generally being equal to $\boldsymbol{p}_{\text{mesh}}$. Though this did not seem to have a noticeable affect on the graphical outcome of the results presented in Section 4.7, we now propose some improvements to the method to overcome this issue. These improvements are a better alternative than the omission of ghost particles due to their crucial role in improving the accuracy of the grid based velocity and normal field.

First, instead of interpolating the ghost particle velocities as the linear combination of the velocities of their parents in (4.51), we need to transfer momentum from the parents to the ghost particles. Second, for the conservation to hold while interpolating back to the mesh, we need $\sum_{p\in b} m_p^n S_i(\boldsymbol{x}_p^n) = m_{bi}^n$ (see (4.82)), which motivates the definition of an improved FLIP-type velocity update to the real particles $r \in R$, where we interpolate momenta back to real as well as ghost particles and then gather them at the respective parents based on the barycentric weights.

Lemma 4.2. *The following changes to the process of ghost particles introduction will reestablish momentum conservation:*

(i) Define the velocity of any ghost particle $q \in G$ as

$$\bar{\boldsymbol{v}}_q^{n+1} = \frac{\sum_{r\in R} \frac{w_r^q m_{r\alpha}}{W_{r\alpha}} \bar{\boldsymbol{v}}_r^{n+1}}{\sum_{r\in R} \frac{w_r^q m_{r\alpha}}{W_{r\alpha}}} \quad \text{with } \alpha \text{ such that } q \in S^\alpha. \tag{4.86}$$

(ii) For any real particles $r \in R$, use the following velocity update:

$$\begin{aligned}\boldsymbol{v}_r^{n+1} = \frac{1}{\sum_\alpha m_{r\alpha}} \Bigg[&\left(\boldsymbol{v}_r^n + \sum_i S_i(\boldsymbol{x}_r^n)(\boldsymbol{v}_{bi}^{n+1} - \boldsymbol{v}_{bi}^n) \right) m_r^n \\ &+ \sum_{\alpha | r\in S^\alpha} \sum_{q\in G\cap S^\alpha} \left(\boldsymbol{v}_q^n + \sum_i S_i(\boldsymbol{x}_q^n)(\boldsymbol{v}_{bi}^{n+1} - \boldsymbol{v}_{bi}^n) \right) \frac{w_r^q m_{r\alpha}}{W_{r\alpha}} \Bigg].\end{aligned} \tag{4.87}$$

Proof. Note that the momentum computations for the transfer to the grid as well as the collision response on the grid do not depend on the addition of the ghost particles, thus these steps remain the same. The mesh momentum after the addition of the ghost particles as well as the mesh momentum after the velocity update need to be recomputed.

(i) The above definition (4.86) conserves momentum on the mesh, since

$$\boldsymbol{p}_{\text{mesh, ghost}} = \sum_{r\in R} m_r^n \bar{\boldsymbol{v}}_r^{n+1} + \sum_{q\in G} m_q^n \bar{\boldsymbol{v}}_q^{n+1} \tag{4.88}$$

$$= \sum_{r\in R} \left(\sum_\alpha \frac{m_{r\alpha}^n}{W_{r\alpha}} \right) \bar{\boldsymbol{v}}_r^{n+1} + \sum_\alpha \sum_{q\in G\cap S^\alpha} \left[\left(\sum_{r\in R} \frac{w_r^q m_{r\alpha}^n}{W_{r\alpha}} \right) \left(\frac{\sum_{r\in R} \frac{w_r^q m_{r\alpha}}{W_{r\alpha}} \bar{\boldsymbol{v}}_r^{n+1}}{\sum_{r\in R} \frac{w_r^q m_{r\alpha}}{W_{r\alpha}}} \right) \right] \tag{4.89}$$

$$= \sum_{r\in R} \left(\sum_\alpha \frac{m_{r\alpha}^n}{W_{r\alpha}} \right) \bar{\boldsymbol{v}}_r^{n+1} + \sum_\alpha \sum_{q\in G\cap S^\alpha} \left[\left(\sum_{r\in R} \frac{w_r^q m_{r\alpha}}{W_{r\alpha}} \bar{\boldsymbol{v}}_r^{n+1} \right) \right] \tag{4.90}$$

$$= \sum_{r\in R} \bar{\boldsymbol{v}}_r^{n+1} \sum_\alpha \left[\frac{m_{r\alpha}^n}{W_{r\alpha}} \underbrace{\Big(1 + \sum_{q\in G\cap S^\alpha} w_r^q \Big)}_{W_{r\alpha}} \right] \tag{4.91}$$

$$= \sum_{r\in R} \bar{\boldsymbol{v}}_r^{n+1} \sum_\alpha m_{r\alpha}^n \tag{4.92}$$

$$= \boldsymbol{p}_{\text{mesh}}, \tag{4.93}$$

(ii) The new total momentum of the mesh with the interpolation defined in (4.87)

reads as (since ghost particles are discarded after the collision step)

$$\boldsymbol{p}_{\text{mesh}}^{\text{FLIP}} = \sum_{r\in R} \boldsymbol{v}_r^{n+1} \sum_{\alpha} m_{r\alpha} \tag{4.94}$$

$$= \sum_{r\in R} \left(\sum_{\alpha} m_{r\alpha} \right) \frac{1}{\sum_{\alpha} m_{r\alpha}} \left[\left(\boldsymbol{v}_r^n + \sum_i S_i(\boldsymbol{x}_r^n)(\boldsymbol{v}_{bi}^{n+1} - \boldsymbol{v}_{bi}^n) \right) m_r^n \right.$$
$$\left. + \sum_{\alpha | r\in S^\alpha} \sum_{q\in G\cap S^\alpha} \left(\boldsymbol{v}_q^n + \sum_i S_i(\boldsymbol{x}_q^n)(\boldsymbol{v}_{bi}^{n+1} - \boldsymbol{v}_{bi}^n) \right) \frac{w_r^q m_{r\alpha}}{W_{r\alpha}} \right]; \tag{4.95}$$

reordering the sums yields

$$\boldsymbol{p}_{\text{mesh}}^{\text{FLIP}} = \sum_{r\in R} \left(\boldsymbol{v}_r^n + \sum_i S_i(\boldsymbol{x}_r^n)(\boldsymbol{v}_{bi}^{n+1} - \boldsymbol{v}_{bi}^n) \right) m_r^n$$
$$+ \sum_{\alpha} \sum_{q\in G\cap S^\alpha} \sum_{r\in S^\alpha} \left(\boldsymbol{v}_q^n + \sum_i S_i(\boldsymbol{x}_q^n)(\boldsymbol{v}_{bi}^{n+1} - \boldsymbol{v}_{bi}^n) \right) \frac{w_r^q m_{r\alpha}}{W_{r\alpha}} \tag{4.96}$$

$$= \sum_{r\in R} \left(\boldsymbol{v}_r^n + \sum_i S_i(\boldsymbol{x}_r^n)(\boldsymbol{v}_{bi}^{n+1} - \boldsymbol{v}_{bi}^n) \right) m_r^n$$
$$+ \sum_{\alpha} \sum_{q\in G\cap S^\alpha} \left(\boldsymbol{v}_q^n + \sum_i S_i(\boldsymbol{x}_q^n)(\boldsymbol{v}_{bi}^{n+1} - \boldsymbol{v}_{bi}^n) \right) \underbrace{\sum_{r\in S^\alpha} \frac{w_r^q m_{r\alpha}}{W_{r\alpha}}}_{=m_q^n}. \tag{4.97}$$

The two remaining sums in the second term cover all ghost particles, thus we can rewrite the right hand side as

$$\boldsymbol{p}_{\text{mesh}}^{\text{FLIP}} = \sum_{r\in R} \left(\boldsymbol{v}_r^n + \sum_i S_i(\boldsymbol{x}_r^n)(\boldsymbol{v}_{bi}^{n+1} - \boldsymbol{v}_{bi}^n) \right) m_r^n$$
$$+ \sum_{q\in G} \left(\boldsymbol{v}_q^n + \sum_i S_i(\boldsymbol{x}_q^n)(\boldsymbol{v}_{bi}^{n+1} - \boldsymbol{v}_{bi}^n) \right) m_q^n \tag{4.98}$$

$$= \sum_{p\in R\cup G} \boldsymbol{v}_p^n m_p^n + \underbrace{\sum_i \underbrace{\sum_{p\in R\cup G} m_p^n S_i(\boldsymbol{x}_q^n)}_{=m_{bi}^n} (\boldsymbol{v}_{bi}^{n+1} - \boldsymbol{v}_{bi}^n)}_{=0 \text{ as shown above}} \tag{4.99}$$

$$= \boldsymbol{p}_{\text{mesh, ghost}} \overset{(4.93)}{=} \boldsymbol{p}_{\text{mesh}}. \tag{4.100}$$

□

4.6 Algorithm

Assembling all the pieces as discussed in the previous sections, our method reads as shown in Algorithm 4.1.

Algorithm 4.1 A Level Set Method for Ductile Fracture

Initialization: create mesh, embed initial geometry, set initial displacements and velocities, boundary conditions, tolerances etc.

for all k **do**
// first, obtain candidate positions and velocities based on the elastic response
$\bar{\boldsymbol{u}}^{n+1} \leftarrow$ fem_elasticity_update$(\boldsymbol{u}^n, \boldsymbol{v}^n, \phi^n)$
$\bar{\boldsymbol{v}}^{n+1} \leftarrow \frac{\bar{\boldsymbol{u}}^{n+1} - \boldsymbol{u}^n}{\Delta t}$

// resolve collisions by way of the Eulerian background grid, using the candidate velocities defined at the old positions
$(\boldsymbol{u}^{n+1}, \boldsymbol{v}^{n+1}) \leftarrow$ collision_handling$(\boldsymbol{u}^n, \bar{\boldsymbol{v}}^{n+1})$

// use the elastic energy densitiy based on the current displacement to advance the material geometry defined by the level set function
$\Psi^{n+1} \leftarrow$ compute_energy_density(u^{n+1})
$\phi^{n+1} \leftarrow$ level_set_update(Ψ^{n+1}, ϕ^n)

// embed the surface of the new healthy region into the mesh; use the old and the new interface to generate new fragments
embedded_geometry(ϕ^{n+1}, ϕ^n)

// reinitialize the level set function to signed distance
compute_exact_distance_in_boundary_elements(ϕ^{n+1})
fast_sweeping(ϕ^{n+1})
end for

4.7 Results

We now demonstrate the compelling, realistic effects that are possible with our method. We show a number of different fracture scenarios, with varying geometric complexity. All examples are computed using $\epsilon = h$ for δ_ϵ in the interface approximation using (2.6), where $h = \Delta x = \Delta y = \Delta z$ is the mesh spacing. The tolerance used in the MINRES solver is set to 10^{-6} and Newton's method solved to 10^{-3}. The time stepping Δt varies with simulation from 10^{-3} to 10^{-5}s. Some representative resolutions, degree of freedom counts and run times can be found in Table 4.1. The material parameters are chosen to emulate real materials; however, sometimes they were adjusted for optical effects, especially the fracture resistance κ. We keep κ constant in each example for

reasons of simplicity, however, spatially varying energy release rates are also possible. All renderings are created using the ray-tracing software *POV-Ray*[2].

First, we show some 2D examples to illustrate our method. Figure 4.4 shows the energy-driven failure evolution while a block with a predefined notch is stretched and tears apart. The material parameters are $E = 0.02$ MPa, $\mu = 0.3$ and $\rho = 1.1\,\frac{\text{kg}}{\text{m}^2}$ and we show an energy release rate $\kappa = 10^5\,\frac{\text{J}}{\text{m}^2}$. Figure 4.5 depicts the sequential generation of fragments of material during a collision-induced failure process of fiberboard with the properties $E = 5$ MPa, $\mu = 0.3$, $\rho = 3.1\,\frac{\text{kg}}{\text{m}^2}$ and $\kappa = 5000\,\frac{\text{J}}{\text{m}^2}$. Our collision handling is further validated with two 3D examples: two rubber cubes ($E = 0.1$ GPa, $\mu = 0.3$, $\rho = 1.1\,\frac{\text{kg}}{\text{m}^3}$) fall onto the ground as well as each other in Figure 4.6; the scalability of the method is displayed in Figure 4.7 with many long rectangular cuboids (slightly softer rubber with E increased to 0.1 GPa) fall onto a partial sphere and the ground.

Figures 4.8, 4.9 and 4.10 depict 3-dimensional failure as a result of stretching. For these tests, we simulate Jell-O$^{\text{TM}}$ as a representative of gelatin and estimate its material parameters as $E = 0.1$ MPa, $\mu = 0.3$ and $\rho = 1.1\,\frac{\text{kg}}{\text{m}^3}$. The fracture resistance in Figure 4.8 is set to $\kappa = 10^5\,\frac{\text{J}}{\text{m}^3}$. Figure 4.9 shows how we can control the speed of the fracture evolution via different energy release rates (note that all three objects are subject to the same external load). Figure 4.10 demonstrates a first example of our ability to fracture complex geometries as we tear off the limbs of an armadillo model. The energy release rate in its arms and legs is given as $\kappa = 3000\,\frac{\text{J}}{\text{m}^3}$. Figure 4.12 shows the collision of two destructible objects. For this example we use fairly hard material with $E = 10$ GPa, $\mu = 0.3$ and $\rho = 1.3\,\frac{\text{kg}}{\text{m}^3}$.

We also demonstrate our fracture response resulting from collisions with external projectiles. The projectile speed is set to $v_{\text{proj}} = 100\frac{\text{m}}{\text{s}}$, thus requiring time steps as small as $\Delta t = 10^{-5}$s. We simulate the projectiles as rigid bodies of analytic shapes and use their exact normals to resolve the collisions between them and the deformable objects (see Section 4.5.5). Figure 4.11 uses the material parameters $E = 5$ MPa, $\mu = 0.3$, $\rho = 1.1\,\frac{\text{kg}}{\text{m}^3}$ and $\kappa = 10^6\,\frac{\text{J}}{\text{m}^3}$ and depicts a bullet through a soft, plastic wall. In Figure 4.13 we shoot a thin sheet with $E = 1.3$ MPa, $\mu = 0.3$, $\rho = 0.5\,\frac{\text{kg}}{\text{m}^3}$ and $\kappa = 1.5 \cdot 10^4\,\frac{\text{J}}{\text{m}^3}$ from the side. In Figure 4.14 we demolish the armadillo, this time with material properties $E = 2$ GPa, $\mu = 0.3$, $\rho = 1.1\,\frac{\text{kg}}{\text{m}^3}$ and $\kappa = 10^6\,\frac{\text{J}}{\text{m}^3}$, by hitting it with two spheres in succession. Lastly, Figure 4.16 presents the projectile penetrating a big block of Jell-O$^{\text{TM}}$ gelatine, using $E = 0.05$ MPa, $\mu = 0.3$, $\rho = 1.1\,\frac{\text{kg}}{\text{m}^3}$ and $\kappa = 10^4\,\frac{\text{J}}{\text{m}^3}$. In the examples in Figure 4.11 and 4.14 we further split damaged fragments using pre-scored grain boundaries as in [6]. This post-processing is restricted to newly formed fragments as determined by our fracture evolution and only used to obtain smaller fragments and is not used to define the damaged region in the first place.

[2]POV-ray (The Persistence of Vision Raytracer): `http://www.povray.org/`

example	dofs	level set grid	min/frame
Stretching Jell-O^{TM}	1.9M	$128 \times 128 \times 128$	4.1
Shooting Jell-O^{TM}	1.0M	$128 \times 128 \times 128$	1.1
Stretching armadillo	1.0M	$96 \times 96 \times 96$	1.1

Table 4.1: Example degree of freedom counts, level set grid resolutions and simulation times. Simulations were performed on a 16-core Intel Xeon E5-2690 2.90GHz machine.

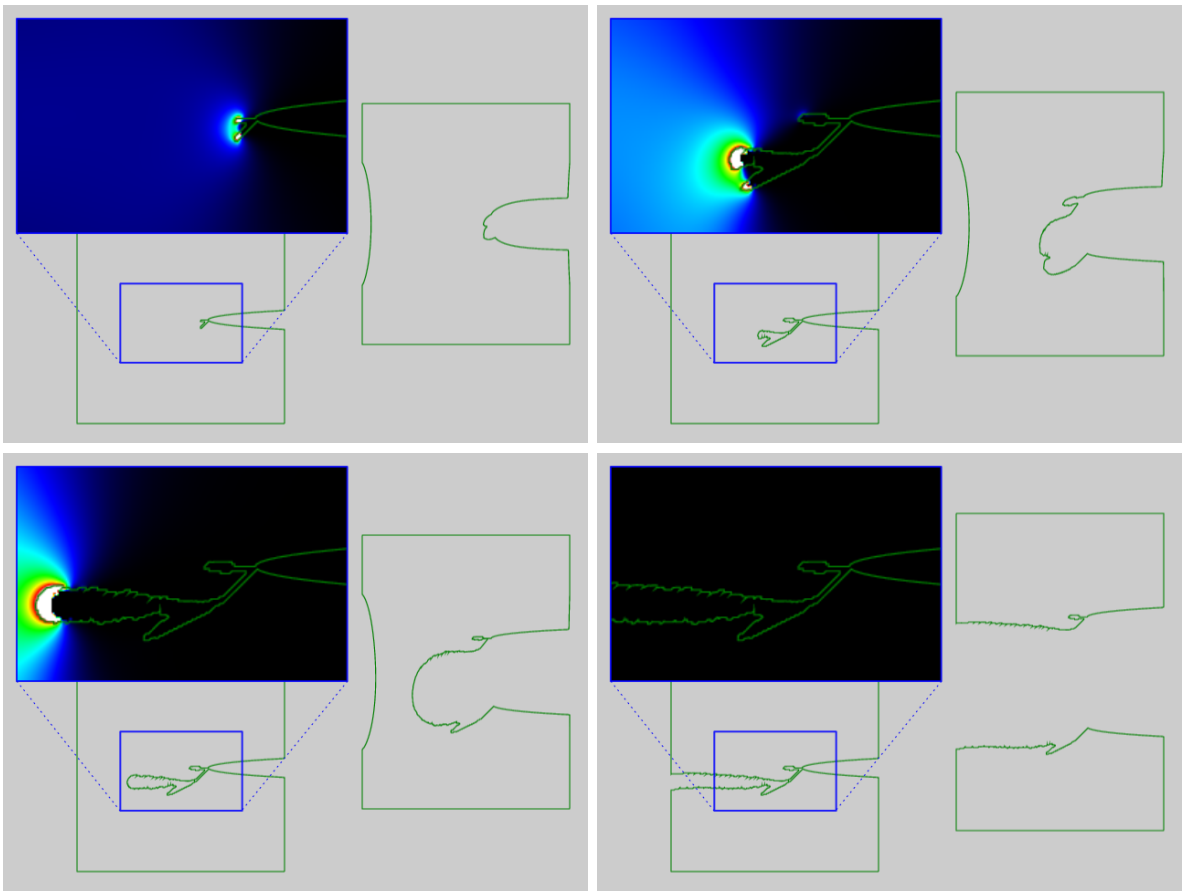


Figure 4.4: Notch tears when stretched. We visualize the material configuration (*bottom left*), the energy density of a subregion (*top left*) and the deformed configuration (*right*).

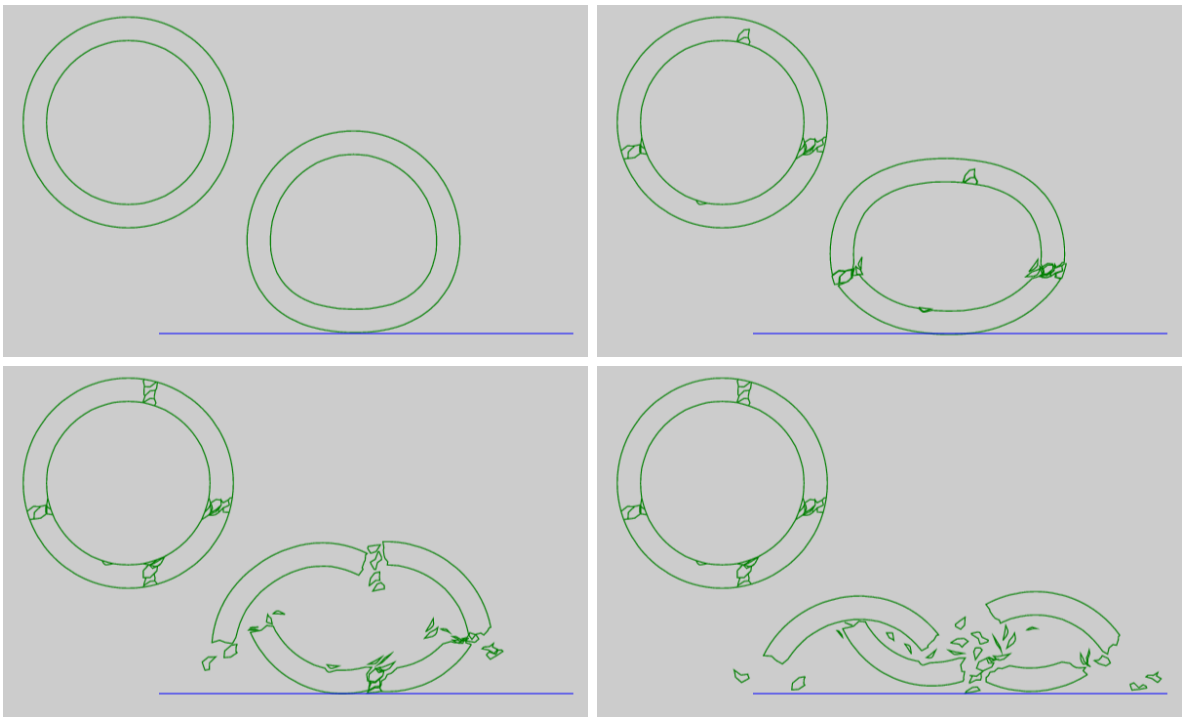


Figure 4.5: Ring fractures upon hitting ground with material configuration (*left*) and the deformed configuration (*right*) shown.

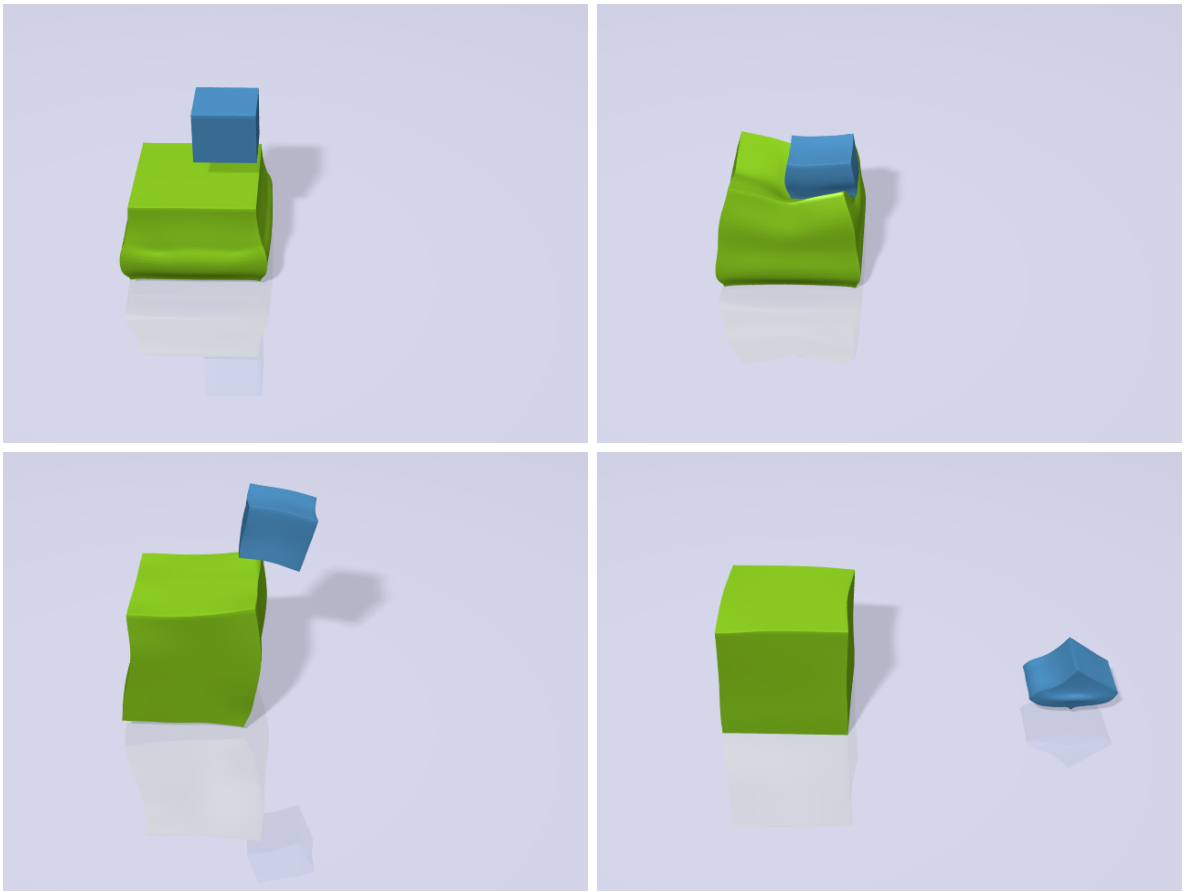


Figure 4.6: Two cubes collide with the ground and each other.

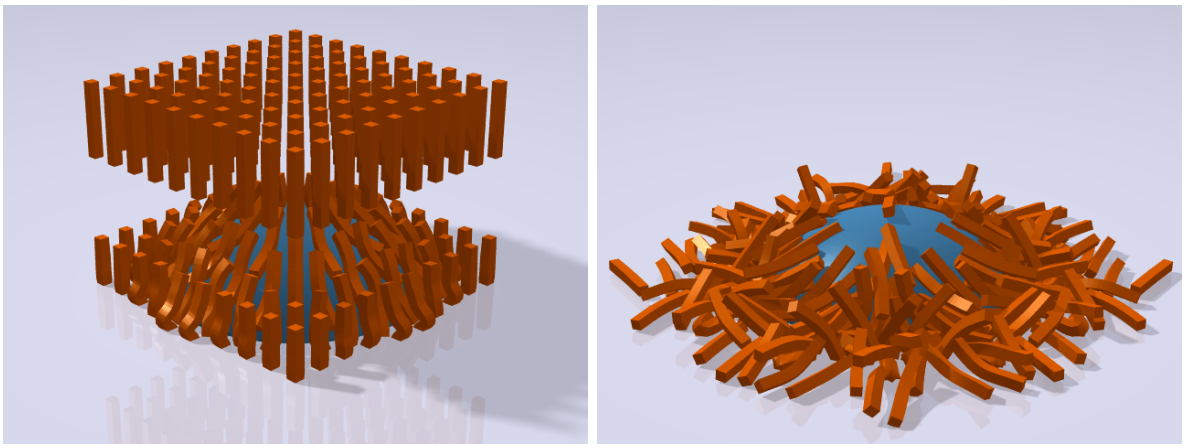
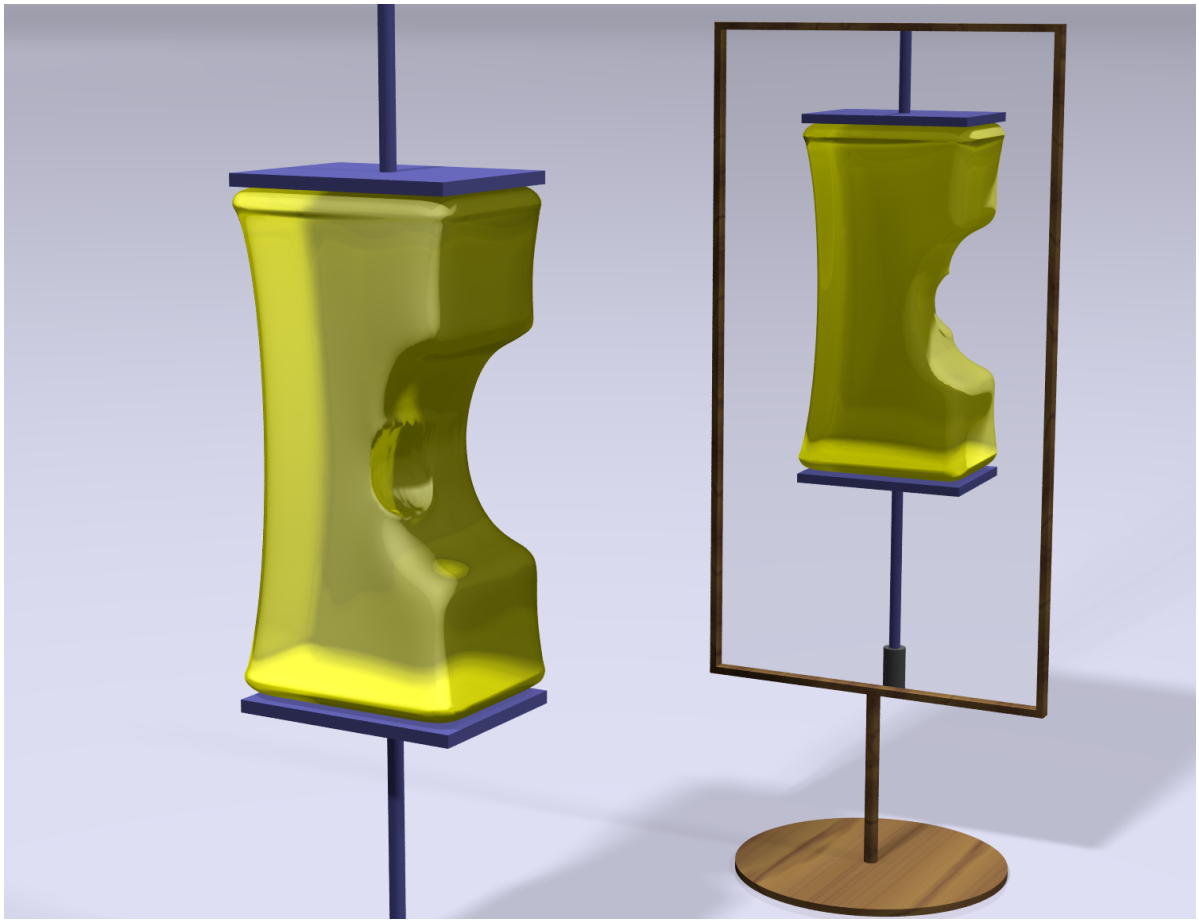
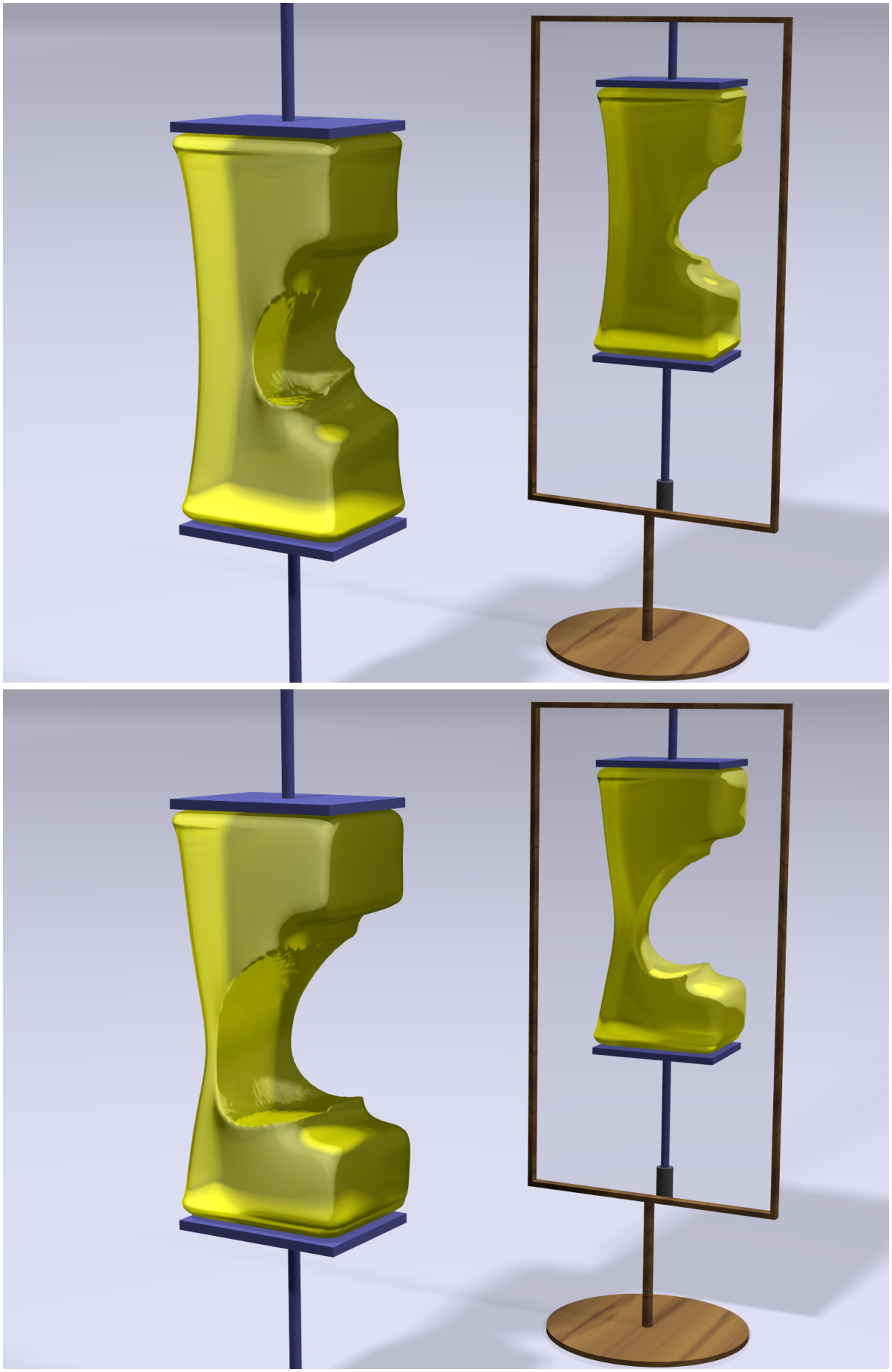


Figure 4.7: Scalability of the collision handling: Many thin pieces fall onto a partial sphere and the ground.





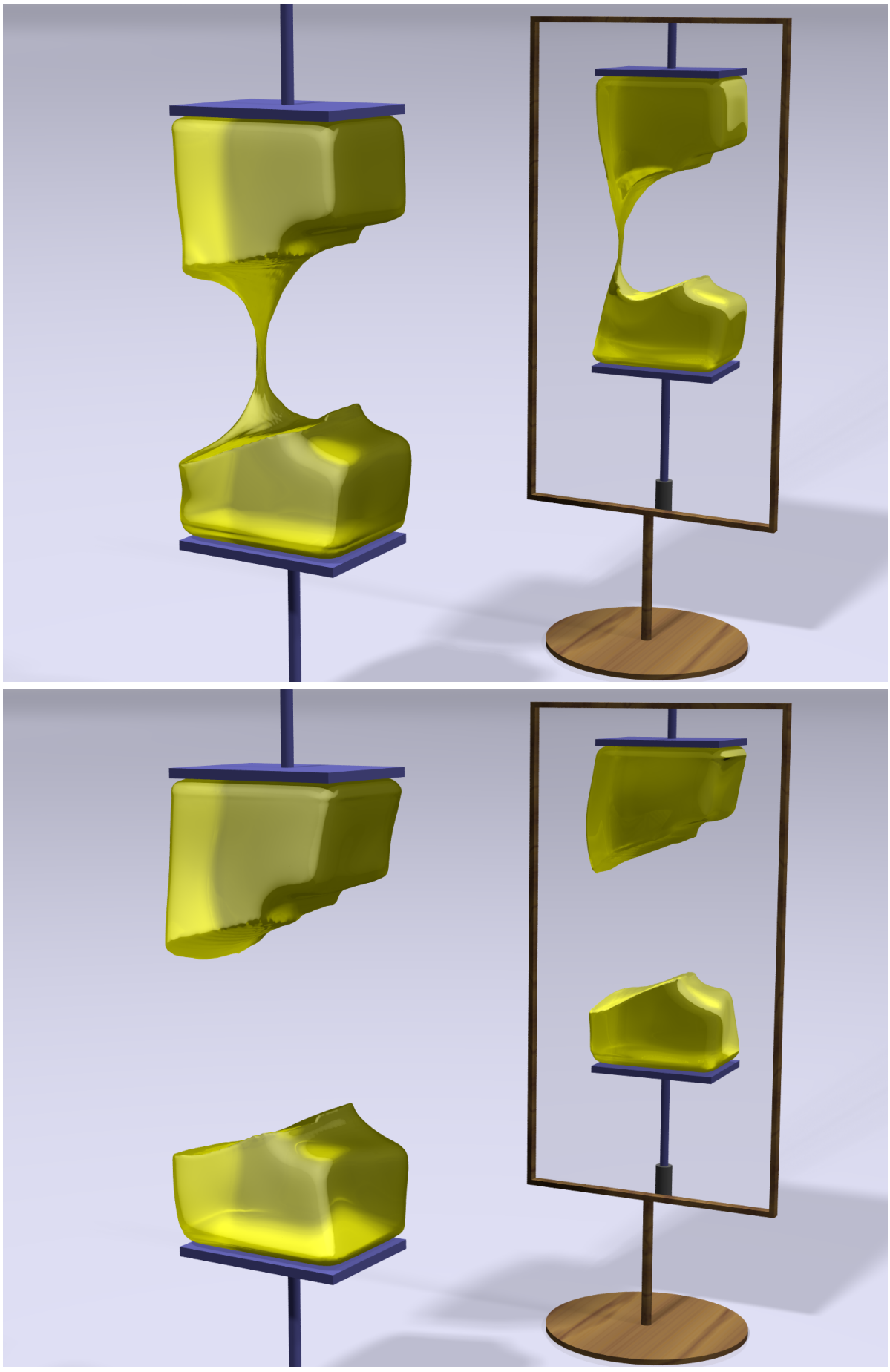


Figure 4.8: A block of Jell-O^{TM} tearing under external load.

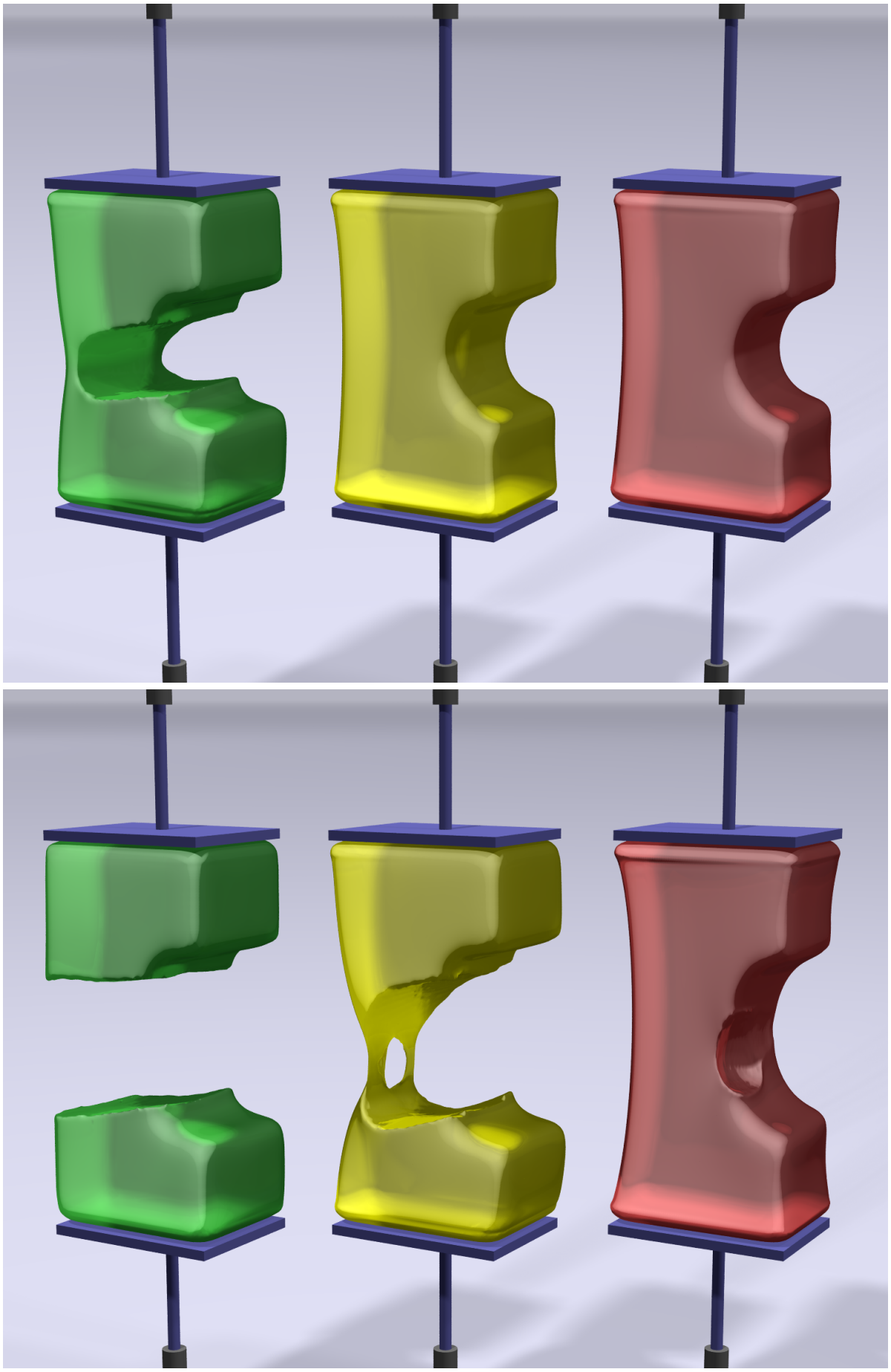
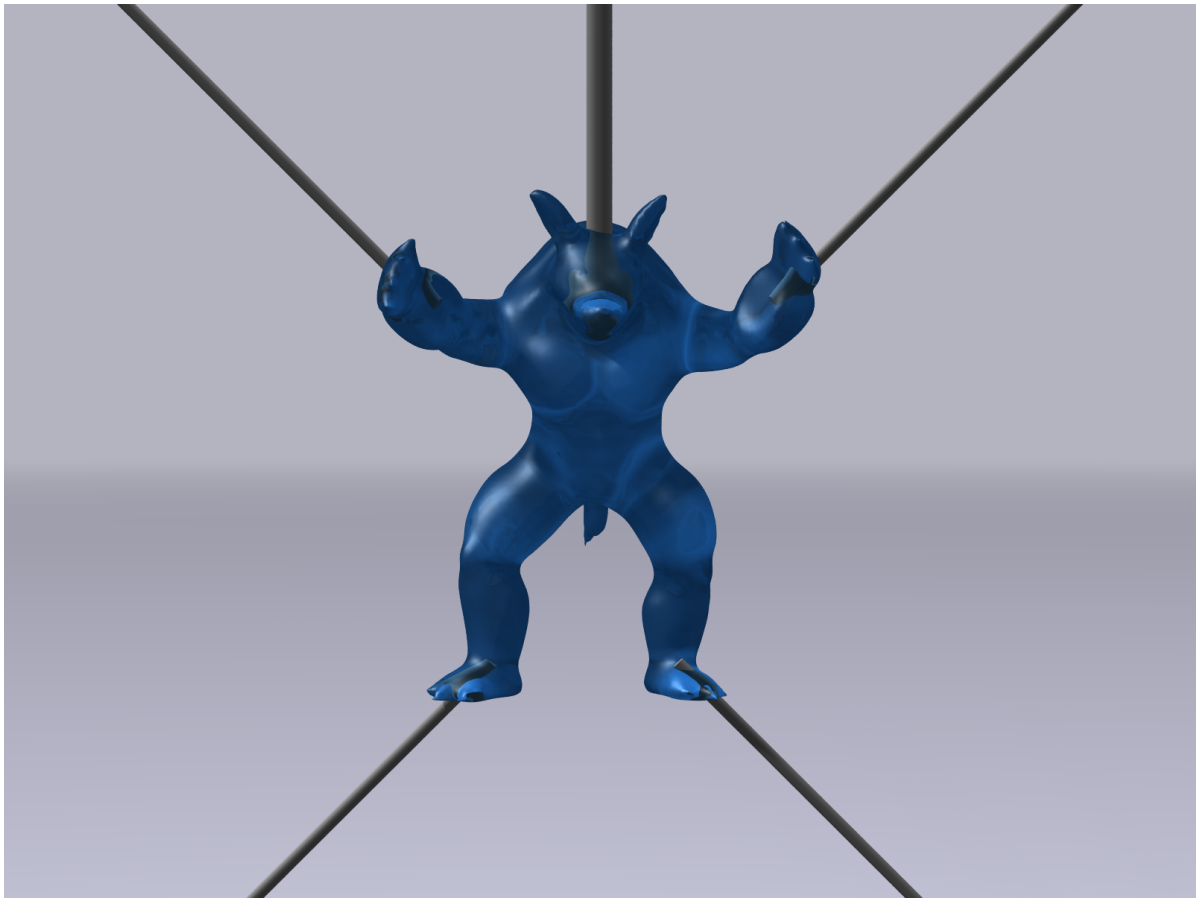
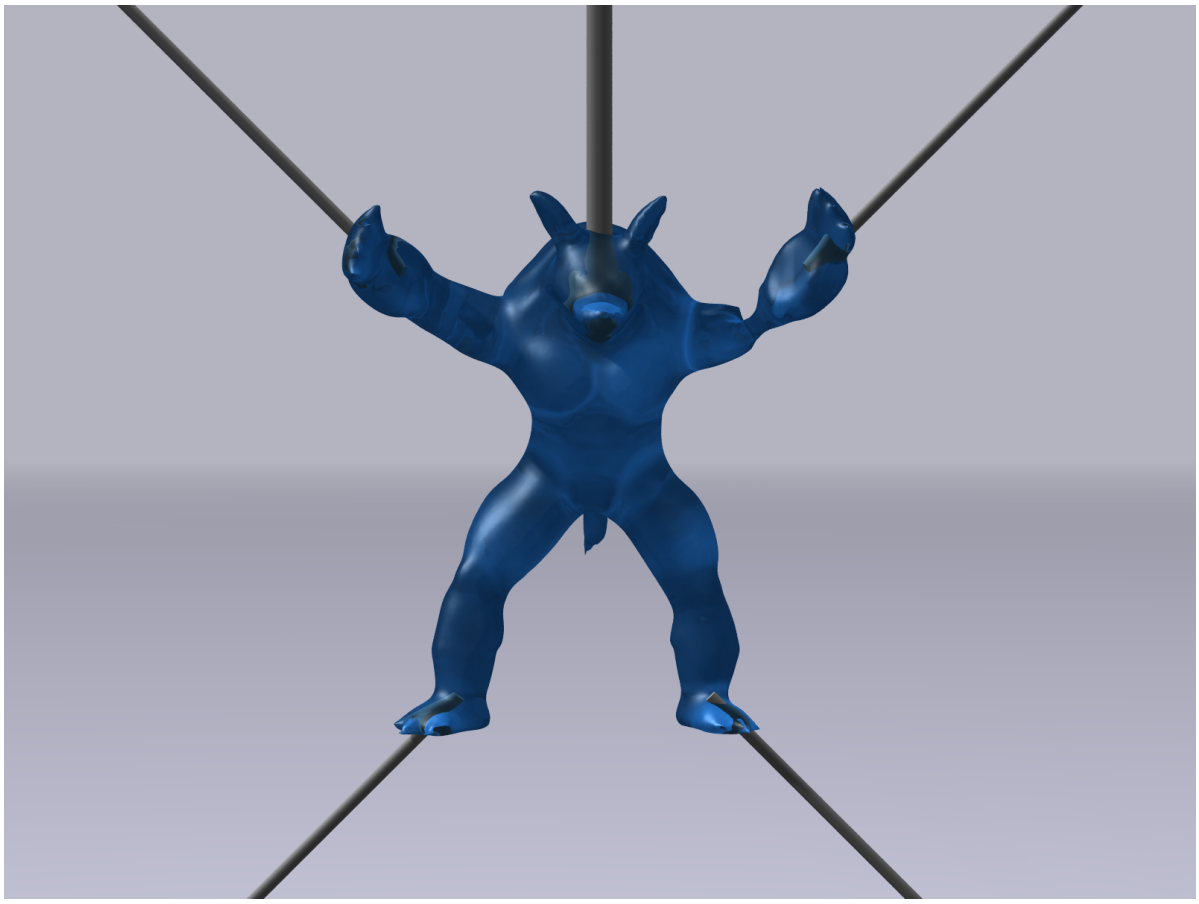
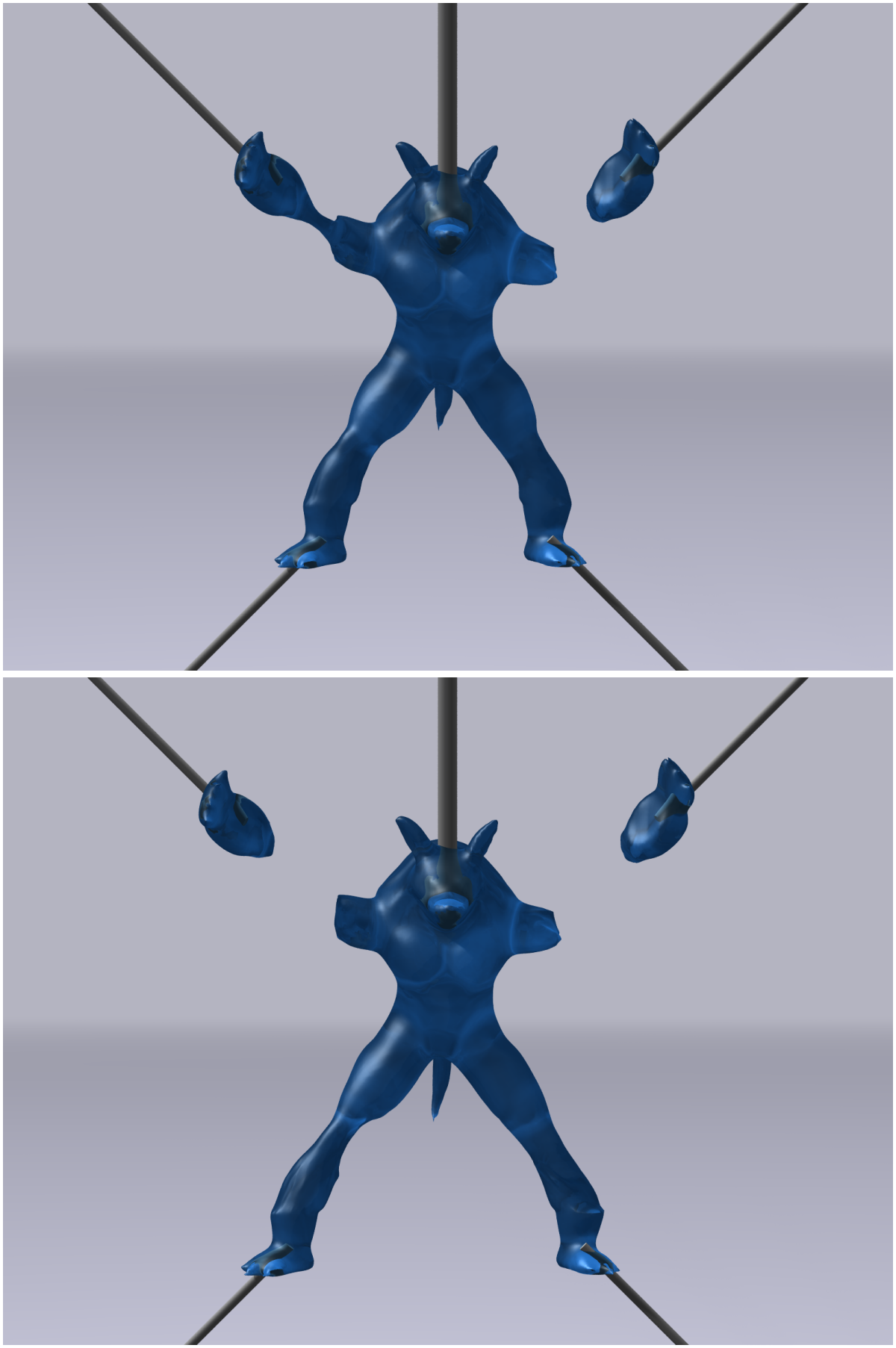


Figure 4.9: Stretching Jell-O^{TM} blocks of different energy release rates, leading to different fracture speeds.







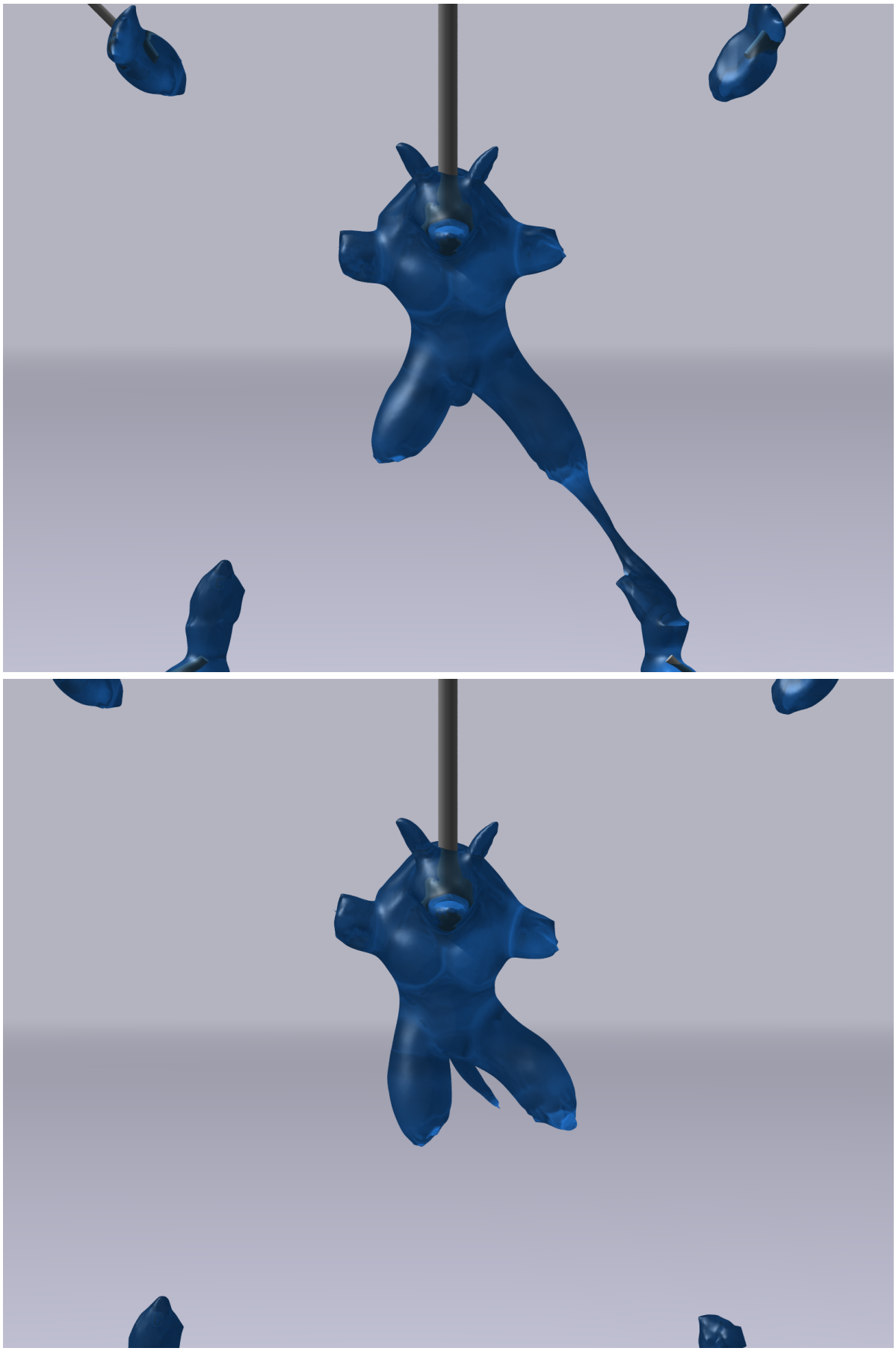


Figure 4.10: An armadillo is stretched until its limbs tear off.

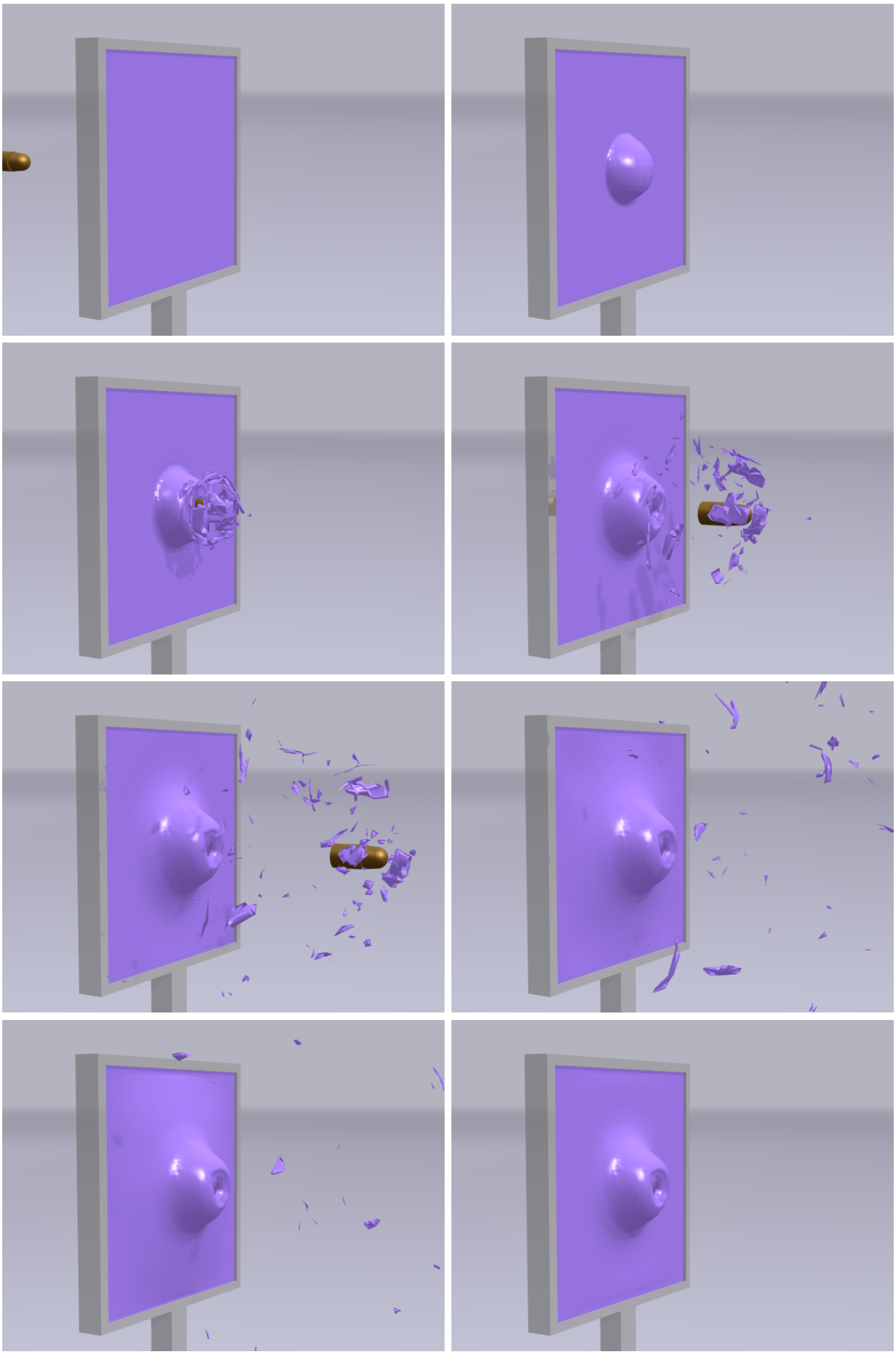


Figure 4.11: Shooting a bullet through a wall, causing irreversible plastic deformation.

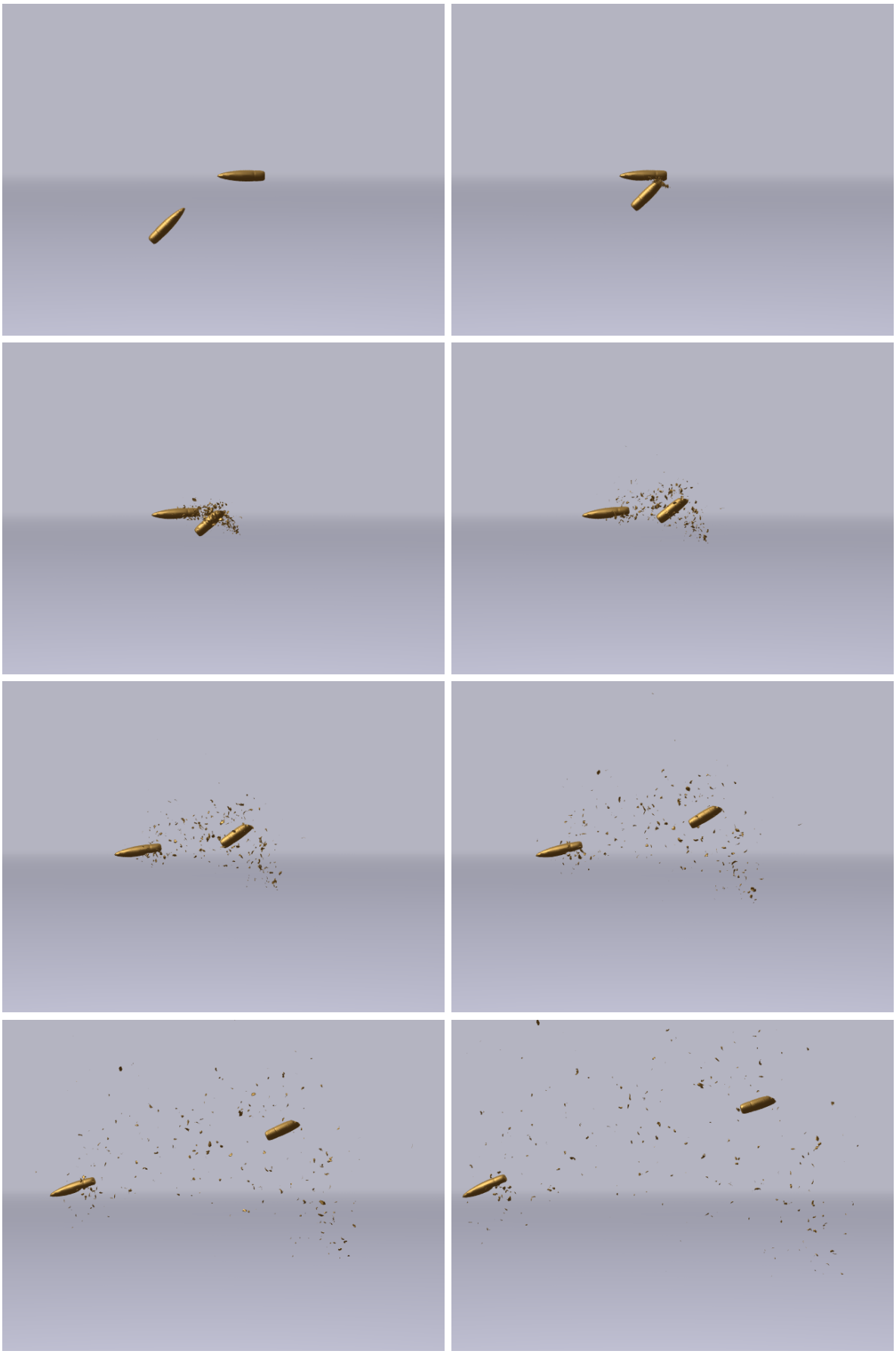
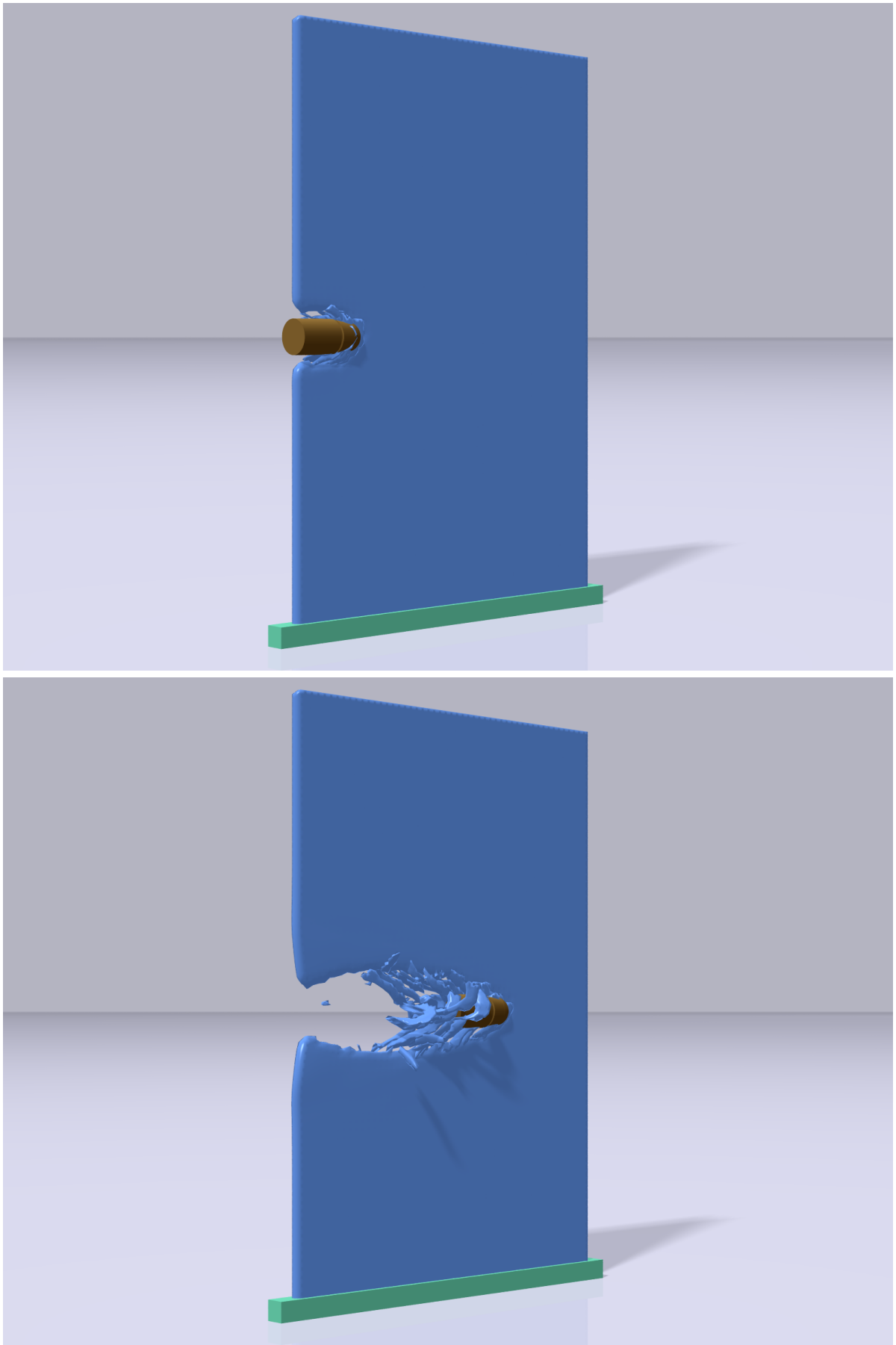


Figure 4.12: Two bullets collide in mid-air and shed small pieces.



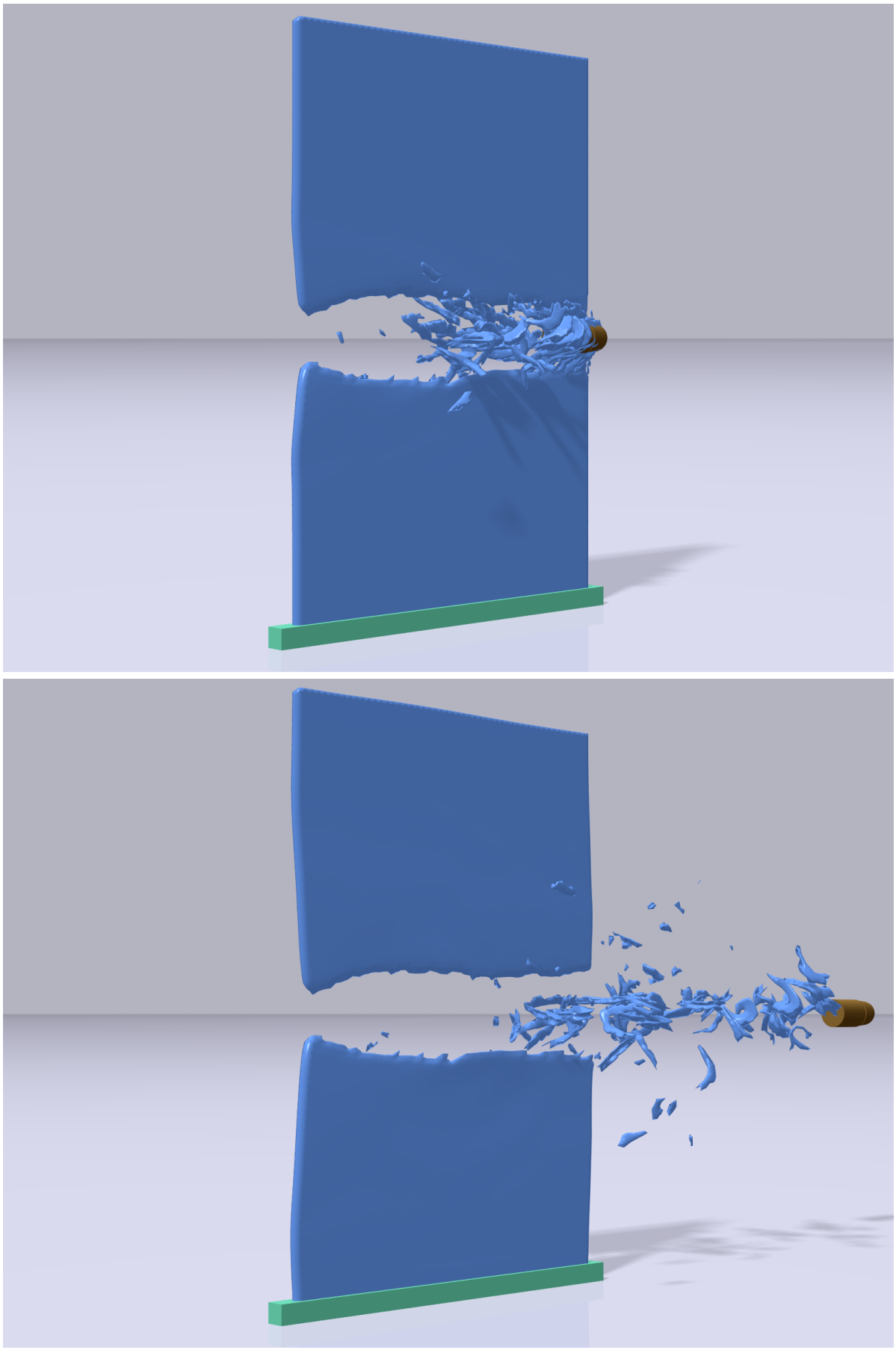
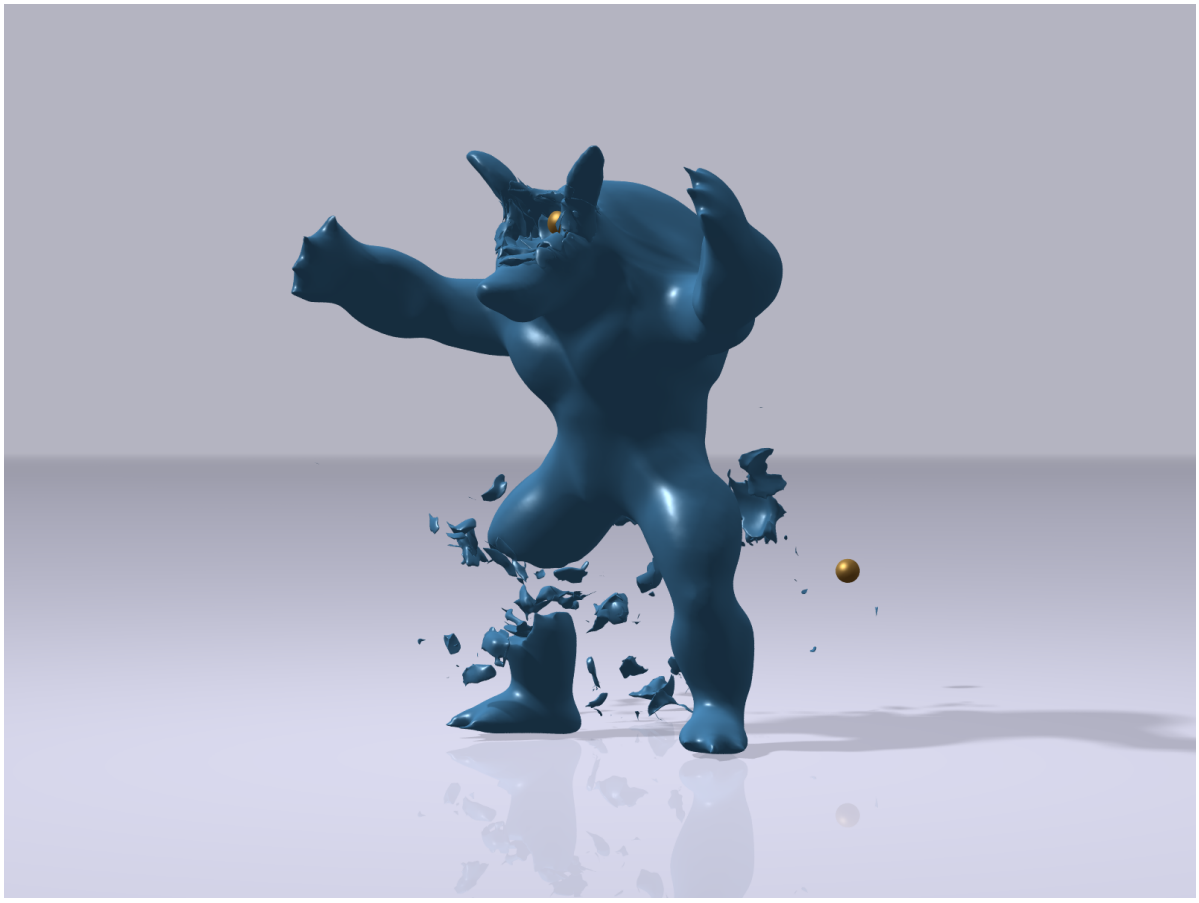


Figure 4.13: We shoot a thin sheet from the side.



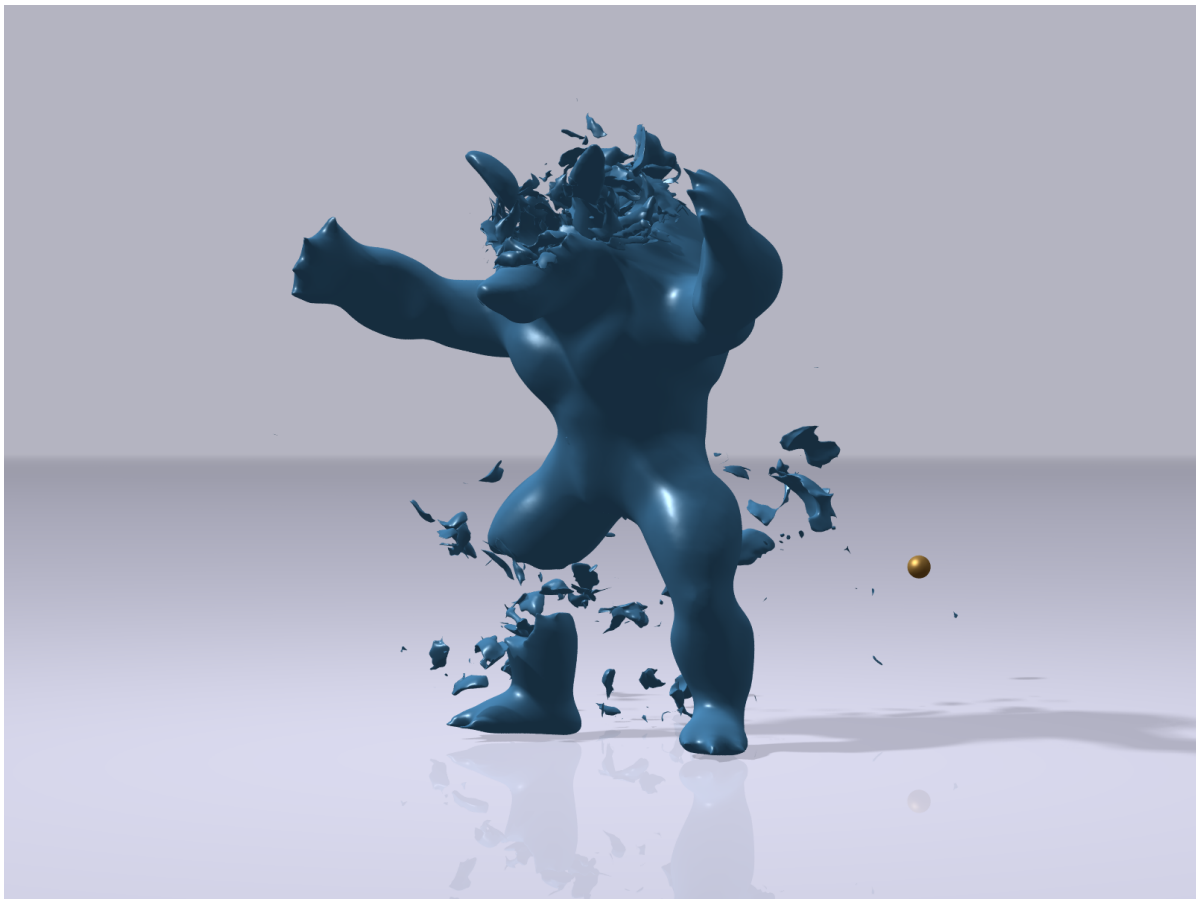


Figure 4.14: Shooting two spheres at an armadillo.

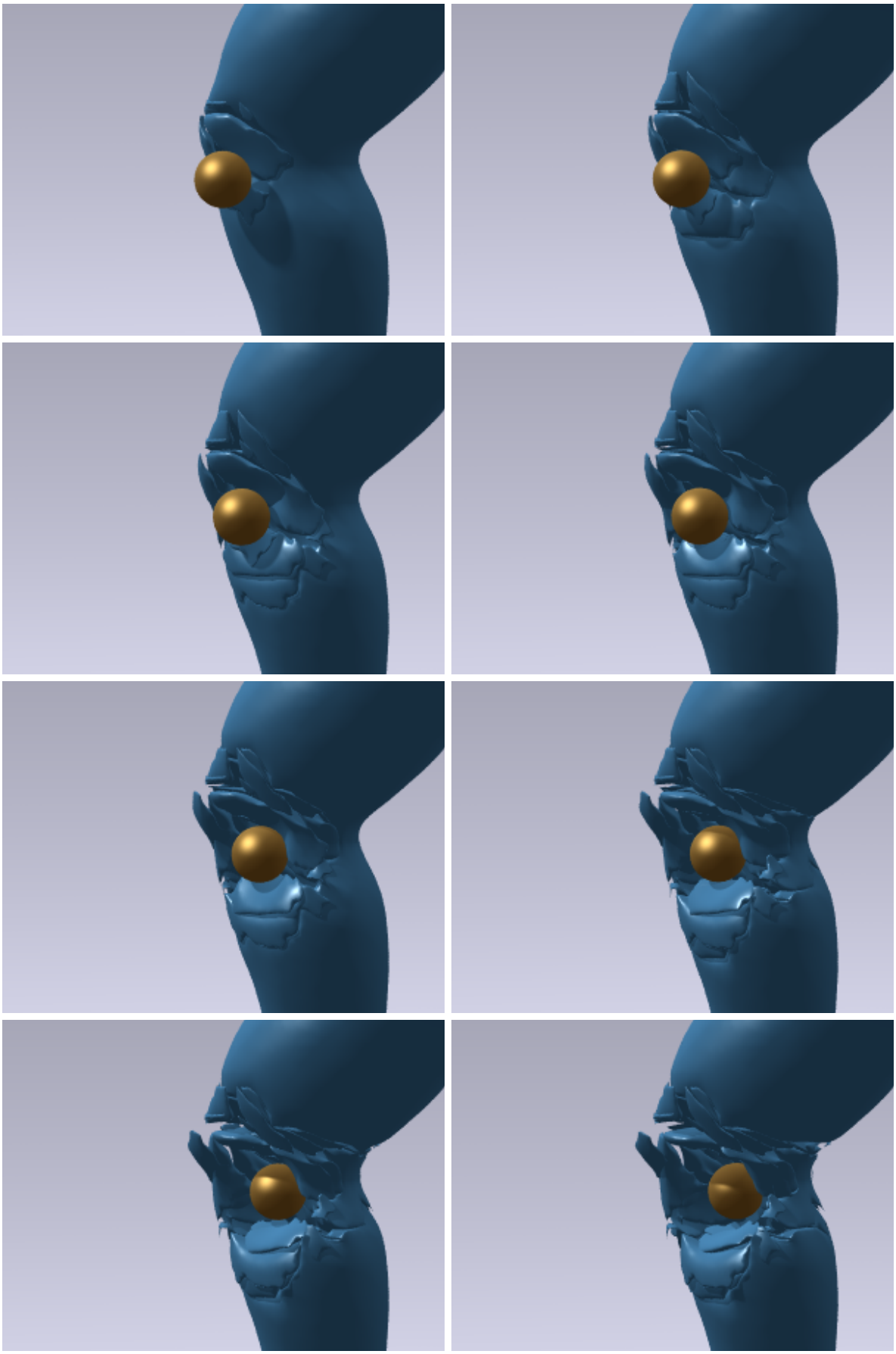
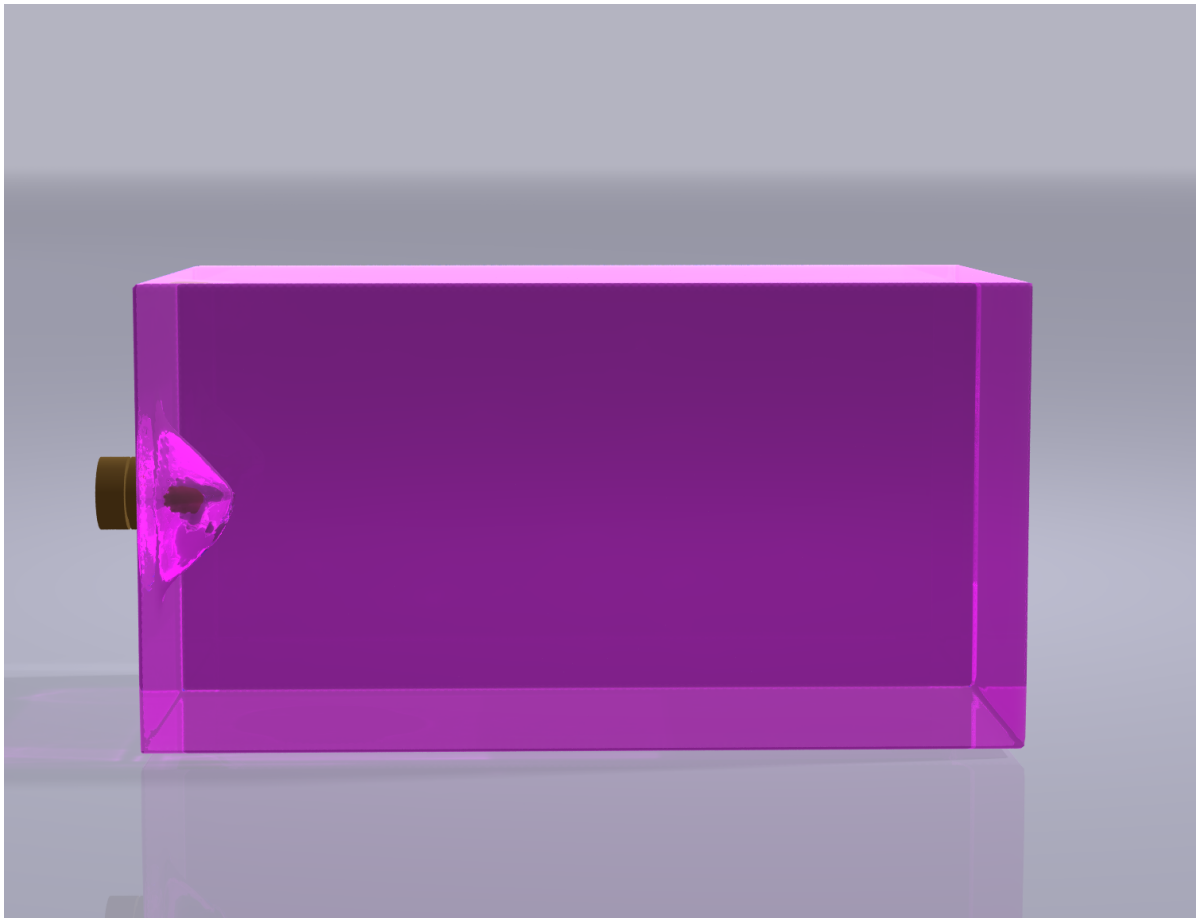
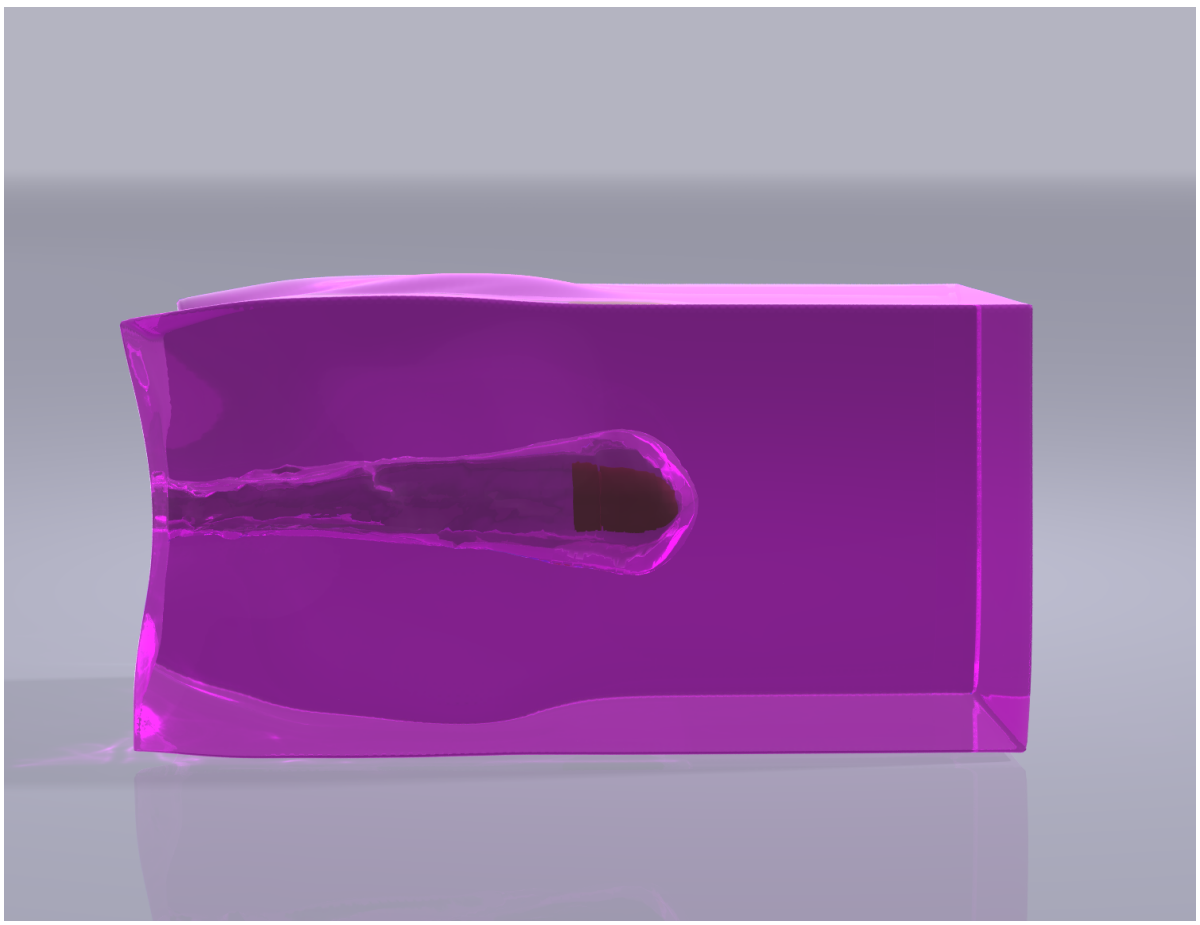


Figure 4.15: Close-up of the fracture response to the impact of an external projectile.





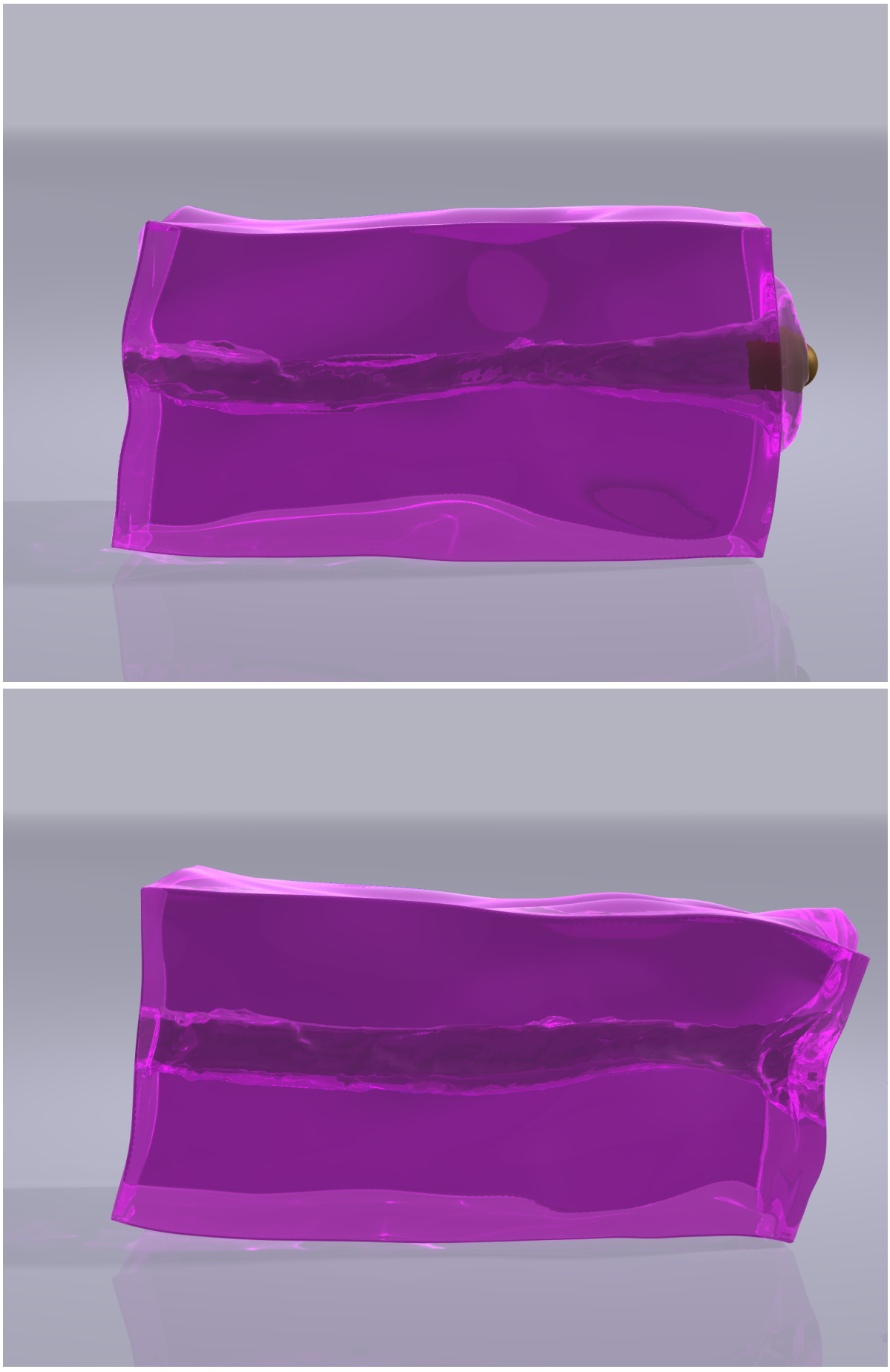


Figure 4.16: Shooting a bullet through Jell-O^{TM}.

4.8 Conclusion

Our level set approach to ductile fracture requires no Lagrangian re-meshing effort and is very easy to implement. However, one drawback of our method is that is suffers from a relative lack of accuracy, in the sense that the level set representation requires a layer of non-zero thickness for the failure region, resulting in an inability to represent individual crack-tip curves. Our grid based collision handling is not as precise as the method of Bridson et al in [20] and can cause small regions of overlap in some areas and separation distances in others. However in contrast to said work, the MPM method provides the capability to resolve collisions between embedded interface, independent of the aspect ratios of the embedded triangles.

Another advantage of our method is that we are not subject to various restrictions observed in existing fracture simulation methods. Our method does not rely on a set of pre-scored fracture paths or even initial cracks and neither are we limited to fracture along element boundaries. We have the ability to propagate at any increment size rather than being limited to an a priori minimum step size or by the mesh element size. Remeshing is not necessary in our method due to the embedding of the material surface into the mesh rather than using interface conforming elements (cf. [105, 104]). Some previous works have also suffered from "back-cracking", where a crack would propagate in a backwards direction in addition to propagating in front of a given crack tip (see Figure 12 in [105]).

In conclusion, our method presents a nice balance between accuracy and complexity of implementation. Its foundation in Griffith's energy and its ability to employ arbitrary fracture patterns lead to compelling, realistic fracture effects, as demonstrated in our results. Furthermore, our level set evolution in the Griffith's energy minimization requires little more information than is already needed during standard simulation of deformable objects. We can also control the fracture process in various ways with our method, e.g. not only can we tune the propagation speed via kappa, but one could also use a spatially varying energy release rate to guide the fracture pattern whenever a more directed evolution is preferred.

Bibliography

[1] G. Allaire, F. Jouve, and A. M. Toader. A level-set method for shape optimization. *Comptes Rendus Mathematique*, 334(12):1125–1130, 2002.

[2] G. Allaire, F. Jouve, and A. M. Toader. Structural optimization using sensitivity analysis and a level-set method. *Journal of Computational Physics*, 194(1):363–393, 2004.

[3] G. Allaire, F. Jouve, and N. Van Goethem. A level set method for the numerical simulation of damage evolution. *Proceedings of ICIAM 2007 Zürich*, pages 3–22, 2009.

[4] D. Alvarez, O. Dorn, and M. Moscoso. A new level-set technique for the crack-detection problem. *PAMM*, 7(1):1081501–1081502, 2007.

[5] U. Ascher and E. Haber. Computational methods for large distributed parameter estimation problems with possible discontinuities. In *Symp. Inverse Problems, Design & Optimization*, pages 201–208, 2004.

[6] Z. Bao, J. M. Hong, J. Teran, and R. Fedkiw. Fracturing rigid materials. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):370–378, 2007.

[7] P. Bastian and C. Engwer. An unfitted finite element method using discontinuous Galerkin. *International Journal for Numerical Methods in Engineering*, 79(12):1557–1576, 2009.

[8] J. Bedrossian, J. H. von Brecht, S. Zhu, E. Sifakis, and J. M. Teran. A second order virtual node method for elliptic problems with interfaces and irregular domains. *Journal of Computational Physics*, 229(18):6405–6426, 2010.

[9] T. Belytschko, R. Gracie, and G. Ventura. A review of extended/generalized finite element methods for material modeling. *Modelling and Simulation in Materials Science and Engineering*, 17(4):043001, 2009.

[10] H. Ben Ameur, M. Burger, and B. Hackl. Level set methods for geometric inverse problems in linear elasticity. *Inverse Problems*, 20:673–696, 2004.

[11] M. P. Bendsøe and O. Sigmund. *Topology optimization: theory, methods, and applications*. Springer, Berlin, 2003.

[12] M. Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

[13] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems.

[14] M. Benzi and A. J. Wathen. Some preconditioning techniques for saddle point problems. In *Model Order Reduction: Theory, Research Aspects and Applications*, pages 195–211. Springer, 2008.

[15] D. Bielser and M. H. Gross. Interactive simulation of surgical cuts. In *Proceedings of the Eighth Pacific Conference on Computer Graphics and Applications*, pages 116–442. IEEE, 2000.

[16] J. Bonet and R. D. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, 2nd edition, 2008.

[17] L. Borcea, G. A. Gray, and Y. Zhang. Variationally constrained numerical solution of electrical impedance tomography. *Inverse Problems*, 19:1159–1184, 2003.

[18] A. Borzì and V. Schulz. Multigrid methods for PDE optimization. *SIAM review*, 51(2):361–395, 2009.

[19] J. U. Brackbill and H. M. Ruppel. Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, 65(2):314–343, 1986.

[20] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 594–603. ACM, 2002.

[21] M. Burger. A level set method for inverse problems. *Inverse problems*, 17:1327, 2001.

[22] M. Burger. A framework for the construction of level set methods for shape optimization and reconstruction. *Interfaces and Free boundaries*, 5(3):301–330, 2003.

[23] M. Burger. Levenberg–Marquardt level set methods for inverse obstacle problems. *Inverse problems*, 20:259–282, 2004.

[24] M. Burger, N. Matevosyan, and M. T. Wolfram. A level set based shape optimization method for an elliptic obstacle problem. *Mathematical Models and Methods in Applied Sciences*, 21(04):619–649, 2011.

[25] M. Burger and S. J. Osher. A survey on level set methods for inverse problems and optimal design. *European Journal of Applied Mathematics*, 16(02):263–301, 2005.

[26] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović. Interactive skeleton-driven dynamic deformations. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 586–593. ACM, 2002.

[27] V. J. Challis, A. P. Roberts, and A. H. Wilkins. Fracture resistance via topology optimization. *Structural and Multidisciplinary Optimization*, 36(3):263–271, 2008.

[28] T. F. Chan and X.-C. Tai. Augmented Lagrangian and total variation methods for recovering discontinuous coefficients from elliptic equations. In *Computational Science for the 21st Century. Symposium*, pages 597–607, 1997.

[29] T. F. Chan and X.-C. Tai. Identification of discontinuous coefficients in elliptic problems using total variation regularization. *SIAM Journal on Scientific Computing*, 25:881–904, 2003.

[30] T. F. Chan and X.-C. Tai. Level set and total variation regularization for elliptic inverse problems with discontinuous coefficients. *Journal of Computational Physics*, 193(1):40–66, 2004.

[31] Y. G. Chen, Y. Giga, and S. Goto. Uniqueness and existence of viscosity solutions of generalized mean curvature flow equations. *Journal of Differential Geometry*, 33(3):749–786, 1991.

[32] Z. Chen and R. Brannon. An evaluation of the material point method. *SAND Report, SAND2002-0482,(February 2002)*, 2002.

[33] Z. Chen and J. Zou. An augmented Lagrangian method for identifying discontinuous parameters in elliptic systems. *SIAM Journal on Control and Optimization*, 37(3):892–910, 1999.

[34] M. Cheney, D. Isaacson, and J. C. Newell. Electrical impedance tomography. *SIAM review*, 41(1):85–101, 1999.

[35] MS Cheung and WC Li. Probabilistic fatigue and fracture analyses of steel bridges. *Structural safety*, 25(3):245–262, 2003.

[36] E. T. Chung, T. F. Chan, and X.-C. Tai. Electrical impedance tomography using level set representation and total variational regularization. *Journal of Computational Physics*, 205(1):357–372, 2005.

[37] B Cotterell. The past, present, and future of fracture mechanics. *Engineering Fracture Mechanics*, 69(5):533–553, 2002.

[38] G.R. Cowper. Gaussian quadrature formulas for triangles. *International Journal for Numerical Methods in Engineering*, 7(3):405–408, 1973.

[39] A. DeCezaro, A. Leitão, and X.-C. Tai. On multiple level-set regularization methods for inverse problems. *Inverse Problems*, 25:035004, 2009.

[40] O. Dorn and D. Lesselier. Level set methods for inverse scattering. *Inverse Problems*, 22(4):R67–R131, 2006.

[41] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*. OUP Oxford, 2005.

[42] M. W. Evans, F. H. Harlow, and E. Bromberg. The particle-in-cell method for hydrodynamic calculations. Technical report, DTIC Document, 1957.

[43] P. Faloutsos, M. Van De Panne, and D. Terzopoulos. Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):201–214, 1997.

[44] C. Forest, H. Delingette, and N. Ayache. Removing tetrahedra from a manifold mesh. In *Computer Animation*, pages 225–229. IEEE, 2002.

[45] G. A. Francfort and A. Garroni. A variational view of partial brittle damage evolution. *Archive for rational mechanics and analysis*, 182(1):125–152, 2006.

[46] G. A. Francfort and J.-J. Marigo. Revisiting brittle fracture as an energy minimization problem. *J. Mech. Phys. Solids, 46: 1319-1342*, 1998.

[47] M. Gissler, M. Becker, and M. Teschner. Constraint sets for topology-changing finite element models. In *Workshop in Virtual Reality Interactions and Physical Simulation*, pages 21–26. The Eurographics Association, 2007.

[48] M. S. Gockenbach, B. Jadamba, and A. A. Khan. Equation error approach for elliptic inverse problems with an application to the identification of Lamé parameters. *Inverse Problems in Science and Engineering*, 16(3):349–367, 2008.

[49] M. S. Gockenbach and A. A. Khan. An abstract framework for elliptic inverse problems: part 1. An output least-squares approach. *Mathematics and Mechanics of Solids*, 12(3):259–276, 2007.

[50] M. S. Gockenbach and A. A. Khan. An abstract framework for elliptic inverse problems: part 2. An augmented Lagrangian approach. *Mathematics and Mechanics of Solids*, 14(6):517–539, 2009.

[51] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

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

[53] A. A. Griffith. The phenomena of rupture and flow in solids. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character, Vol. 221: 163-198*, 1921.

[54] J. Hegemann. An adaptive hybrid method for 2D crack growth simulation. Diploma thesis, Westfälische Wilhelms-Universität Münster, 2011. Available online at `http://wwwmath.uni-muenster.de/num/burger/teaching/diplomanden.html`.

[55] J. Hegemann, A. Cantarero, C. L. Richardson, and J. M. Teran. An explicit update scheme for inverse parameter and interface estimation of piecewise constant coefficients in linear elliptic PDEs. *SIAM Journal on Scientific Computing*, 35(2):A1098–A1119, 2013.

[56] J. Hegemann, C. Jiang, C. Schroeder, and J. M. Teran. A level set method for ductile fracture. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 2013. Accepted.

[57] J. Hellrung, A. Selle, A. Shek, E. Sifakis, and J. Teran. Geometric fracture modeling in Bolt. In *SIGGRAPH 2009: Talks*, page 7:1. ACM, 2009.

[58] J. L. Hellrung Jr, L. Wang, E. Sifakis, and J. M. Teran. A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions. *Journal of Computational Physics*, 231(4):2015–2048, 2012.

[59] F. Hettlich and W. Rundell. Iterative methods for the reconstruction of an inverse potential problem. *Inverse Problems*, 12:251–266, 1996.

[60] F. Hettlich and W. Rundell. Recovery of the support of a source term in an elliptic differential equation. *Inverse Problems*, 13:959–976, 1997.

[61] F. Hettlich and W. Rundell. The determination of a discontinuity in a conductivity from a single boundary measurement. *Inverse Problems*, 14:67–82, 1998.

[62] C. Hogea, C. Davatzikos, and G. Biros. An image-driven parameter estimation problem for a reaction–diffusion glioma growth model with mass effects. *Journal of Mathematical Biology*, 56(6):793–825, 2008.

[63] A. Hoger and B. E. Johnson. Linear elasticity for constrained materials: Incompressibility. *Journal of Elasticity*, 38(1):69–93, 1995.

[64] P. Huang, X. Zhang, S. Ma, and X. Huang. Contact algorithms for the material point method in impact and penetration simulation. *International Journal for Numerical Methods in Engineering*, 85(4):498–517, 2011.

[65] H. N. Iben and J. F. O'Brien. Generating surface crack patterns. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 177–185. The Eurographics Association, 2006.

[66] H. N. Iben and J. F. O'Brien. Generating surface crack patterns. *Graphical Models*, 71(6):198–208, 2009.

[67] G. Irving, J. Teran, and R. Fedkiw. Invertible finite elements for robust simulation of large deformation. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 131–140. Eurographics Association, 2004.

[68] K. Ito and K. Kunisch. The augmented Lagrangian method for parameter estimation in elliptic systems. *SIAM Journal on Control and Optimization*, 28:113, 1990.

[69] K. Ito, K. Kunisch, and Z. Li. Level-set function approach to an inverse interface problem. *Inverse problems*, 17:1225, 2001.

[70] B. Jadamba, A. A. Khan, and F. Raciti. On the inverse problem of identifying Lamé coefficients in linear elasticity. *Computers & Mathematics with Applications*, 56(2):431–443, 2008.

[71] L. Ji, J. R. McLaughlin, D. Renzi, and J. R. Yoon. Interior elastodynamics inverse problems: shear wave speed reconstruction in transient elastography. *Inverse Problems*, 19:S1, 2003.

[72] P. Kaufmann, S. Martin, M. Botsch, E. Grinspun, and M. Gross. Enrichment textures for detailed cutting of shells. 28(3):50, 2009.

[73] I. Knowles. Parameter identification for elliptic problems. *Journal of Computational and Applied Mathematics*, 131(1-2):175–194, 2001.

[74] I. Knowles, T. Le, and A. Yan. On the recovery of multiple flow parameters from transient head data. *Journal of Computational and Applied Mathematics*, 169(1):1–15, 2004.

[75] V. Kolehmainen, S.R. Arridge, W. R. B. Lionheart, M. Vauhkonen, and J. P. Kaipio. Recovery of region boundaries of piecewise constant coefficients of an elliptic PDE from boundary data. *Inverse Problems*, 15(5):1375–1391, 1999.

[76] K. Kunisch and X. Pan. Estimation of interfaces from boundary measurements. *SIAM Journal on Control and Optimization*, 32:1643–1674, 1994.

[77] K. Kunisch and X.-C. Tai. Sequential and parallel splitting methods for bilinear control problems in Hilbert spaces. *SIAM Journal on Numerical Analysis*, 34(1):91–118, 1997.

[78] Maximilien E Launey and Robert O Ritchie. On the fracture toughness of advanced materials. *Advanced Materials*, 21(20):2103–2110, 2009.

[79] H. S. Lee, C. J. Park, and H. W. Park. Identification of geometric shapes and material properties of inclusions in two-dimensional finite bodies by boundary parameterization. *Computer Methods in Applied Mechanics and Engineering*, 181(1-3):1–20, 2000.

[80] R. J. Leveque and Z. Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM Journal on Numerical Analysis*, 31(4):1019–1044, 1994.

[81] A. Litman, D. Lesselier, and F. Santosa. Reconstruction of a two-dimensional binary obstacle by controlled evolution of a level-set. *Inverse Problems*, 14:685–706, 1998.

[82] 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.

[83] Z. Luo, L. Tong, J. Luo, P. Wei, and M. Y. Wang. Design of piezoelectric actuators using a multiphase level set method of piecewise constants. *Journal of Computational Physics*, 228(7):2643–2659, 2009.

[84] O. Mazarak, C. Martins, and J. Amanatides. Animating exploding objects. In *Proceedings of the 1999 conference on Graphics interface '99*, pages 211–218. Morgan Kaufmann Publishers Inc., 1999.

[85] A. McAdams, S. Osher, and J. Teran. Crashing waves, awesome explosions, turbulent smoke, and beyond: Applied mathematics and scientific computing in the visual effects industry. *AMS Notices*, 57(5):614–623, 2010.

[86] A. McAdams, A. Selle, R. Tamstorf, J. Teran, and E. Sifakis. Computing the singular value decomposition of 3x3 matrices with minimal branching and elementary floating point operations. Technical report, University of Wisconsin - Madison, 2011. Available online at `http://pages.cs.wisc.edu/~sifakis/`.

[87] A. McAdams, E. Sifakis, and J. Teran. A parallel multigrid Poisson solver for fluids simulation on large grids. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 65–74. Eurographics Association, 2010.

[88] A. McAdams, Y. Zhu, Andrew Selle, Mark Empey, R. Tamstorf, J. Teran, and E. Sifakis. Efficient elasticity for character skinning with contact and collisions. In *ACM Transactions on Graphics (TOG)*, volume 30, page 37. ACM, 2011.

[89] J. McLaughlin and D. Renzi. Using level set based inversion of arrival times to recover shear wave speed in transient elastography and supersonic imaging. *Inverse Problems*, 22:707–725, 2006.

[90] J. R. McLaughlin, N. Zhang, and A. Manduca. Calculating tissue shear modulus and pressure by 2D log-elastographic methods. *Inverse Problems*, 26:085007, 2010.

[91] B. Merriman, J. K. Bence, and S. J. Osher. Motion of multiple junctions: A level set approach. *Journal of Computational Physics*, 112(2):334–363, 1994.

[92] C. Min and F. Gibou. Geometric integration over irregular domains with application to level-set methods. *Journal of Computational Physics*, 226(2):1432–1443, 2007.

[93] Neil Molino, Zhaosheng Bao, and Ron Fedkiw. A virtual node algorithm for changing mesh topology during simulation. In *ACM Transactions on Graphics (TOG)*, pages 385–392, 2004.

[94] A. B. Mor and T. Kanade. Modifying soft tissue models: Progressive cutting with minimal new element creation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2000*, pages 598–607. Springer, 2000.

[95] M. Müller and M. Gross. Interactive virtual materials. In *Proceedings of Graphics Interface 2004*, pages 239–246. Canadian Human-Computer Communications Society, 2004.

[96] M. Müller, L. McMillan, J. Dorsey, and R. Jagnow. Real-time simulation of deformation and fracture of stiff materials. In *Computer Animation and Simulation*, pages 113–124. Springer, 2001.

[97] F. Natterer. Imaging and inverse problems of partial differential equations. *Preprint (University Münster, 2006)*.

[98] M. Neff and E. Fiume. A visual model for blast waves and fracture. In *Proceedings of the 1999 conference on Graphics interface '99*, pages 193–202. Morgan Kaufmann Publishers Inc., 1999.

[99] Boris Nesterenko, Grirory I Nesterenko, and Valentin N Basov. Fracture behaviour of skin materials of civil airplane structures. In *ICAF 2009, Bridging the Gap between Theory and Operational Practice*, pages 661–683. Springer, 2009.

[100] L. K. Nielsen, X.-C. Tai, S. I. Aanonsen, and M. Espedal. A binary level set model for elliptic inverse problems with discontinuous coefficients. *International Journal of Numerical Analysis and Modeling*, 4(1):74–99, 2007.

[101] A. Norton, G. Turk, B. Bacon, J. Gerth, and P. Sweeney. Animation of fracture by physical modeling. *The Visual Computer*, 7(4):210–219, 1991.

[102] A. A. Oberai, N. H. Gokhale, and G. R. Feijóo. Solution of inverse problems in elasticity imaging using the adjoint method. *Inverse Problems*, 19:297–313, 2003.

[103] J. F. O'Brien, A. W. Bargteil, and J. K. Hodgins. Graphical modeling and animation of ductile fracture. *ACM Transactions on Graphics (TOG)*, 21(3):291–294, 2002.

[104] J. F. O'Brien, A. W. Bargteil, and J. K. Hodgins. Graphical modeling and animation of ductile fracture. *ACM Transactions on Graphics (TOG)*, 21(3):291–294, 2002.

[105] J. F. O'Brien and J. K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 137–146. ACM Press/Addison-Wesley Publishing Co., 1999.

[106] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, New York, 2002.

[107] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[108] S. J. Osher and F. Santosa. Level set methods for optimization problems involving geometry and constraints I. Frequencies of a two-density inhomogeneous drum. *Journal of Computational Physics*, 171(1):272–288, 2001.

[109] C. C. Paige, B. N. Parlett, and H. A. Van der Vorst. Approximate solutions and eigenvalue bounds from Krylov subspaces. *Numerical Linear Algebra with Applications*, 2(2):115–133, 1995.

[110] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[111] E. G. Parker and J. F. O'Brien. Real-time deformation and fracture in a game environment. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 165–175, 2009.

[112] M. Pauly, R. Keiser, B. Adams, P. Dutré, M. Gross, and L. J. Guibas. Meshless animation of fracturing solids. 24(3):957–964, 2005.

[113] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A PDE-based fast local level set method. *Journal of Computational Physics*, 155(2):410–438, 1999.

[114] C. Ramananjaona, M. Lambert, and D. Lesselier. Shape inversion from TM and TE real data by controlled evolution of level sets. *Inverse Problems*, 17:1585–1595, 2001.

[115] C. Ramananjaona, M. Lambert, D. Lesselier, and J. P. Zolésio. Shape reconstruction of buried obstacles by controlled evolution of a level set: from a min-max formulation to numerical experimentation. *Inverse Problems*, 17:1087–1111, 2001.

[116] C. Ramananjaona, M. Lambert, D. Lesselier, and J. P. Zolésio. On novel developments of controlled evolution of level sets in the field of inverse shape problems. *Radio Science*, 37(2):8010, 2002.

[117] Krishnaswamy Ravi-Chandar. *Dynamic fracture*. Elsevier Science, 2004.

[118] T. Rees, H. S. Dollar, and A. J. Wathen. Optimal solvers for PDE-constrained optimization. *SIAM Journal on Scientific Computing*, 32(1):271–298, 2010.

[119] C. L. Richardson, J. Hegemann, E. Sifakis, J. Hellrung, and J. M. Teran. An xfem method for modeling geometrically elaborate crack propagation in brittle materials. *International Journal for Numerical Methods in Engineering*, 88(10):1042–1065, 2011.

[120] Y. Saad. *Iterative methods for sparse linear systems*, volume 620. PWS publishing company Boston, 1996.

[121] A. P. Santhanam, Y. Min, S. P. Mudur, A. Rastogi, B. H. Ruddy, A. Shah, E. Divo, A. Kassab, J. P. Rolland, and P. Kupelian. An inverse hyper-spherical harmonics-based formulation for reconstructing 3D volumetric lung deformations. *Comptes Rendus Mécanique*, 2010.

[122] F. Santosa. A level-set approach for inverse problems involving obstacles. *ESAIM: Control, Optimization and Calculus of Variations*, 1:17–33, 1996.

[123] D. S. Schnur and N. Zabaras. An inverse method for determining elastic material properties and a material interface. *International Journal for Numerical Methods in Engineering*, 33(10):2039–2057, 1992.

[124] J. Schöberl and W. Zulehner. Symmetric indefinite preconditioners for saddle point problems with applications to PDE-constrained optimization problems. *SIAM Journal on Matrix Analysis and Applications*, 29(3):752–773, 2007.

[125] T.W. Sederberg and S.R. Parry. Free-form deformation of solid geometric models. In *ACM Siggraph Computer Graphics*, volume 20, pages 151–160. ACM, 1986.

[126] J. A. Sethian and A. Wiegmann. Structural boundary design via level set and immersed interface methods. *Journal of Computational Physics*, 163(2):489–528, 2000.

[127] E. Sifakis and J. Barbic. Fem simulation of 3d deformable solids: a practitioner's guide to theory, discretization and model reduction. In *ACM SIGGRAPH Courses*, pages 20:1–20:50. ACM, 2012.

[128] E. Sifakis, K. G. Der, and R. Fedkiw. Arbitrary cutting of deformable tetrahedralized objects. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 73–80. Eurographics Association, 2007.

[129] J. Smith, A. Witkin, and D. Baraff. Fast and controllable simulation of the shattering of brittle objects. 20(2):81–91, 2001.

[130] N. C. Smith and K. Vozoff. Two-dimensional DC resistivity inversion for dipole-dipole data. *IEEE Transactions on Geoscience and Remote Sensing*, GE-22(1):21–28, 1984.

[131] J. Sokolowski and J. P. Zolesio. *Introduction to Shape Optimization: Shape Sensitivity Analysis*. Springer, Berlin, 1992.

[132] M. Soleimani, W. R. B. Lionheart, and O. Dorn. Level set reconstruction of conductivity and permittivity from boundary electrical measurements using experimental data. *Inverse problems in science and engineering*, 14(2):193–210, 2006.

[133] A. Stomakhin, R. Howes, C. Schroeder, and J. M. Teran. Energetically consistent invertible elasticity. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation (SCA)*, pages 25–32. The Eurographics Association, 2012.

[134] A. Stomakhin, R. Howes, C. Schroeder, and J. M. Teran. Energetically consistent invertible elasticity: Supplementary technical document. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation (SCA)*, pages 25–32. The Eurographics Association, 2012. Available online at `http://www.math.ucla.edu/~jteran/`.

[135] J. Su, C. Schroeder, and R. Fedkiw. Energy stability and fracture for frame rate rigid body simulations. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 155–164. ACM, 2009.

[136] D. Sulsky, S. J. Zhou, and H. L. Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, 87(1):236–252, 1995.

[137] M. Sussman, P. Smereka, and S. Osher. *A level set approach for computing solutions to incompressible two-phase flow*, volume 14. 1994.

[138] X.-C. Tai and T. F. Chan. A survey on multiple level set methods with applications for identifying piecewise constant functions. *International Journal of Numerical Analysis and Modelling*, 1(1):25–48, 2004.

[139] X.-C. Tai and H. Li. A piecewise constant level set method for elliptic inverse problems. *Applied Numerical Mathematics*, 57(5-7):686–696, 2007.

[140] J. Teran, E. Sifakis, S.S. Blemker, V. Ng-Thow-Hing, C. Lau, and R. Fedkiw. Creating and simulating skeletal muscle from the visible human data set. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):317–328, 2005.

[141] J. M. Teran, J. L. Hellrung, and J. Hegemann. Simulation of elasticity, biomechanics, and virtual surgery. *IAS/Park City Mathematics Series*, 2012. To appear. Preprint available online at `http://wwwmath.uni-muenster.de/num/burger/organization/hegemannj/`.

[142] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *ACM Siggraph Computer Graphics*, volume 22, pages 269–278. ACM, 1988.

[143] K. Van den Doel and U. M. Ascher. On level set regularization for highly ill-posed distributed parameter estimation problems. *Journal of Computational Physics*, 216(2):707–723, 2006.

[144] K. van den Doel, U. M. Ascher, and A. Leitao. Multiple level sets for piecewise constant surface reconstruction in highly ill-posed problems. *Journal of Scientific Computing*, 43(1):44–66, 2010.

[145] H. A. Van der Vorst. *Iterative Krylov methods for large linear systems*, volume 13. Cambridge University Press, 2003.

[146] B. L. Vaughan, B. G. Smith, and D. L. Chopp. A comparison of the extended finite element method with the immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *Applied Mathematics and Computational Science*, 1(1):207–228, 2006.

[147] L. A. Vese and T. F. Chan. A multiphase level set framework for image segmentation using the Mumford and Shah model. *International Journal of Computer Vision*, 50(3):271–293, 2002.

[148] P. Wei and M. Y. Wang. Piecewise constant level set method for structural topology optimization. *International Journal for Numerical Methods in Engineering*, 78(4):379–402, 2009.

[149] H. Zhao. A fast sweeping method for eikonal equations. *Mathematics of Computation*, 74(250):603–628, 2005.

[150] C. Zheng and D. L. James. Rigid-body fracture sound with precomputed soundbanks. *ACM Transactions on Graphics (TOG)*, 29(4):69, 2010.

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

[152] J. Zou. Numerical methods for elliptic inverse problems. *International Journal of Computer Mathematics*, 70(2):211–232, 1998.

Jan Hegemann

Persönliche Daten

geboren am	30.07.1984
in	Soest

Ausbildung

Promotion (Dr. rer. nat.) in Mathematik

Seit 09/2009	am Institut für Numerische und Angewandte Mathematik der Westfälischen Wilhelms-Universität Münster
Dissertation	*"Efficient Evolution Algorithms for Embedded Interfaces: From Inverse Parameter Estimation to a Level Set Method for Ductile Fracture"*
Betreuer	Prof. Dr. Martin Burger

Diplomstudium in Mathematik

10/2004 - 08/2009	an der Westfälischen Wilhelms-Universität Münster
13/08/2009	Diplomprüfung, bestanden mit Auszeichnung (Note: 1,0)
Diplomarbeit	*"An Adaptive Hybrid Method for 2D Crack Growth Simulation"*
Betreuer	Prof. Dr. Martin Burger

Schulausbildung

27/06/2003	Allgemeine Hochschulreife (Abitur), bestanden mit Durchschnitt 1,3
08/1994 - 06/2003	Aldegrever-Gymnasium Soest
08/1990 - 07/1994	Petri-Grundschule Soest

Berufliche und akademische Erfahrung

Seit 01/2013	Wissenschaftlicher Mitarbeiter an der Westfälischen Wilhelms-Universität Münster
09/2009 - 12/2012	Wissenschaftlicher Mitarbeiter an der University of California, Los Angeles, USA
04/2008 - 04/2009	Gastwissenschaftler an der University of California, Los Angeles, USA
10/2007 - 03/2008	Studentische Hilfskraft an der Westfälischen Wilhelms-Universität Münster

Münster, 16/05/2013