Interfaces and Boundaries in Physics Based Simulation of Solids and Fluids

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von MSc Computational Engineering

Dan Alexander Koschier

aus Offenbach am Main, Deutschland

Berichter: Univ.-Prof. Dr. rer. nat. Jan Bender
Prof. Dr.-Ing. Nils Thuerey

Tag der mündlichen Prüfung: 10.07.2018

Abstract

Recent developments concerning the numerical simulation of solid objects and fluid flows in the field of computer graphics have opened up a plethora of new possibilities in applications such as special effect productions, animated movies, *Virtual Reality* (VR) applications, medical simulators, and computer games. Despite various techniques for the simulation of solids and fluid flows exist, the accurate incorporation of complex boundary geometries and interface models still poses a great challenge. Nevertheless, a robust handling of these interface descriptions is inevitable for a wide range of applications. Besides other purposes, interface models are frequently used to represent the boundary surfaces of solid objects, the layer between different materials, the boundary geometry of domains trapping a fluid, or even to represent cuts, tears, or cracks separating the material within an object. In order to be able to simulate even more complex phenomena and to enhance existing approaches, advanced methods and new techniques for the efficient and robust numerical simulation of solids and fluids with complex interfaces have to be developed.

The contributions of this thesis are organized into three parts. In the first part, two novel methods based on *Finite Element* (FE) discretizations for the simulation of brittle fracture and cutting of deformable solids are presented. The first chapter in this part focuses on the physically motivated generation of brittle fractures using an adaptive stress analysis. While this approach captures crack interfaces by explicit remeshing and element duplication, the approach described in the second chapter of the first part captures the interface implicitly by using enrichment functions that are directly embedded into the FE discretization. The enrichment based technique is able to capture even highly complex and finely structured cuts with high accuracy while any form of remeshing is completely avoided.

The second part of this thesis is concerned with a novel discretization approach for implicit interface representations. An arbitrary surface in three-dimensional space can be represented as an isosurface of a signed distance function. In the first step, the novel approach discretizes the signed distance function into a grid structure using piecewise polynomials. Subsequently, the initial discretization is refined in order to improve the discretization accuracy. The presented method is the first approach that not only refines the grid cells spatially but also varies the degree of the polynomial basis. With this approach even highly complicated surfaces can be accurately discretized while keeping the memory consumption to a minimum.

In the third and final part of this thesis a novel approach for the simulation of incompressible fluids and a method to handle non-penetration boundary conditions using the novel concept of precomputed density maps are presented. Building on the Navier-Stokes equations for isothermal incompressible fluids the partial differential equation is spatially discretized using the *Smoothed Particle Hydrodynamics* (SPH) formalism. Incompressibility is then ensured using a novel pressure solver that enforces both a constant density field throughout the fluid and a divergence-free velocity field. In order to enforce non-penetration an implicit representation of the boundary interface is constructed and a density map is precomputed. Using the novel concept of density maps, non-penetration boundary conditions can be handled using efficient lookups into the map with constant complexity while the requirement to sample the boundary interface geometry with particles vanishes.

Zusammenfassung

Aktuelle Methoden zur numerischen Simulation von Festkörpern und Fluiden in der Computergrafik, eröffnen eine Vielzahl neuer Möglichkeiten für Anwendungen wie bspw. Spezialeffekten in Spielfilmen, vollanimierten Filmen, VR-Anwendungen (VR := Virtuelle Realität), medizinischen Simulatoren und Computerspielen. Obwohl bereits viele Techniken zur Simulation starrer und deformierbarer Festkörper und Fluidströmungen existieren, stellt das Problem der Berücksichtigung komplexer Berandungsgeometrien und Grenzflächenmodellen mit hoher Genauigkeit noch immer eine große Herausforderung dar. Nichtsdestotrotz ist die Verwendung robuster Verfahren zur Berücksichtigung der Grenzflächen für eine Vielzahl von Anwendungen unabdingbar. Neben anderen Verwendungen, werden Grenzflächenmodelle häufig zur Repräsentation der Berandungsfläche von Festkörpern, der Schicht zwischen aneinandergrenzenden Materialien, der Berandung von Behältern, die eine Flüssigkeit einschließen, oder sogar zur Repräsentation von Schnitten, Rissen oder Brüchen, die das Material im Innern eines Körpers separieren, eingesetzt. Um eine robuste Simulation noch komplexerer Phänomene zu ermöglichen und bestehende Ansätze zu erweitern, besteht die Notwendigkeit verbesserte Methoden und neue Techniken zur effizienten und robusten numerischen Simulation von Festkörpern und Fluiden zu entwickeln.

Die wissenschaftlichen Beiträge dieser Dissertation sind in drei Teile gegliedert. Im ersten Teil werden zwei neue Methoden basierend auf FE-Diskretisierungen (FE := Finite Elemente) zur Simulation spröder Brüche und dem Zerschneiden deformierbarer Festkörper vorgestellt. Die erste Arbeit beschäftigt sich mit der physikalisch-motivierten Generierung spröder Brüche mit Hilfe einer neuen adaptiven Spannungsanalyse. Während diese erste Methode die Schnittflächengeometrie durch eine explizite Neuvernetzung und Elementverdopplung erfasst, wird die Schnittfläche im zweiten Ansatz durch eine direkte Einbettung sog. Enrichmentfunktionen in die FE-Diskretisierung implizit erfasst. Mit Hilfe der Enrichment-basierten Technik können hochkomplexe Schnittflächen und feinstrukturierte Schnitte exakt erfasst und robust simuliert werden, während jegliche Form der Neuvernetzung vermieden wird.

Der zweite Teil dieser Arbeit beschäftigt sich mit einem neuen Diskretisierungsansatz für implizite Repräsentationen von Grenzflächenmodellen. Eine beliebige Fläche im dreidimensionalen Raum kann durch eine Isofläche einer vorzeichenbehafteten Distanzfunktionen repräsentiert werden. Der neue Ansatz diskretisiert diese Distanzfunktion zunächst mit Hilfe eines Gitters unter Verwendung abschnittsweise definierter Polynome. Anschließend wird die initiale Diskretisierung verfeinert, um die Diskretisierungsgenauigkeit zu verbessern. Die neue Diskretierungsmethode ist der erste existierende Ansatz, in dem Gitterzellen nicht nur räumlich unterteilt werden, sondern auch der Grad der polynomiellen Basis lokal variiert wird. Mit Hilfe dieses Ansatzes können hochkomplexe Flächen mit hoher Genauigkeit diskretisiert werden, während der Speicherverbrauch minimal gehalten wird.

Im dritten und letzten Teil dieser Dissertation werden ein neuer Ansatz zur Simulation inkompressibler Fluide und eine neue Methode zur Behandlung von Durchdringungsrandbedingungen unter Verwendung des neuen Konzepts der vorberechneten Dichtekarten vorgestellt. Aufbauend auf der Navier-Stokes Gleichung für isothermale, incompressible Fluide, wird die partielle Differentialgleichung mit Hilfe des SPH (SPH := Smoothed Particle Hydrodynamics) Formalismus räumlich diskretisiert. Die Bewahrung der Inkompressbilität wird weiterhin durch einen neuen Drucklöser gewährleistet, der sowohl einen konstantes Dichtefeld als auch ein divergenzfreies Geschwindigkeitsfeld im gesamten Fluid erzwingt. Um eine Durchdringung des Fluids an der Berandung zu gewährleisten, wird eine implizite Repräsentation der Berandung konstruiert und eine sog. Dichtekarte vorberechnet. Mit Hilfe des neuen Konzepts der Dichtekarten können Durchdringungsrandbedingungen unter Verwendung effizienter Zugriffe konstanter Komplexität in die Dichtekarten gehandhabt werden, ohne dabei die Randgeometrie mit Partikeln samplen zu müssen.

Acknowledgements

I would like to gratefully thank all the people who supported me through the time as PhD student and who made this thesis possible in the first place. First and foremost, I would like to sincerely thank my advisor Prof. Dr. Jan Bender. As early as in the third year of my undergraduate studies he enabled me to actively participate in his group's research activities in the field of physically based simulation that built the basis for this thesis. Shortly after completing my undergraduate studies he gave me the opportunity to join the group as a PhD candidate by providing me with a fast-track scholarship even before my postgraduate studies were completed. Moreover, I would like to express my sincerest gratitude for his guidance, time-commitment and motivation to support my efforts as well as the countless hours of fruitful discussions that lead to the here presented work. I would also like to specially thank Prof. Dr. Nils Thürey for his support and engagement in our collaborations.

Moreover, I would like to mention my former colleagues and coauthors of my publications, Sebastian Lipponer, Crispin Deul, Patrick Charrier, Daniel Weber, Marcel Weiler, Magnus Brand and Tassilo Kugelstadt. Thank you for the collaborations and all the help.

I am enormously thankful to my girlfriend Caroline for her love, support, and patience over so many years as well as for her constructive criticism and proofreading concerning this manuscript. Thank you, for putting up with me through the many submission deadlines and for the help to reflect on the important things in life. Additionally, I would like to thank her family, in particular Barbara, Brigitte, and Roland, for all their support.

Special thanks also go to my close friends Nicolas, Daniel, and Fabian for being always there to discuss life choices. Also the time I spent with you as a student, including the many controversial discussions, made my life much more enjoyable than it had been without you.

Finally, I would like to thank my family and, especially, my parents Birgit and Wolfgang. Without your support and sacrifice, it would have never been possible to get the stability in life and the education that I was so lucky to receive which helped me to reach this goal. Also, thanks go to my brothers Neil and Joel and to my grandfathers Herbert and Willi.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

ADF	Adaptive Distance Field
BSH	Bounding Sphere Hierarchy
BSP	Binary Space Partitioning
BVH	Bounding Volume Hierarchy
BVP	Boundary Value Problem
CFL	Courant-Friedrichs-Levy
CPU	Central Processing Unit
DFSPH	Divergence-free Smoothed Particle Hydrodynamics
DOF	Degree of Freedom
EOS	Equation of State
FD	Finite Difference
FE	Finite Element
FEM	Finite Element Method
FVM	Finite Volume Method
GB	Gigabyte
GHz	Gigahertz
GPU	Graphics Processing Unit
IBVP	Mixed Initial-Boundary Value Problem
IISPH	Implicit Incompressible SPH
MLCP	Mixed Linear Complementarity Problem
MLS	Moving Least-Squares
NSE	Navier-Stokes Equations
ODE	Ordinary Differential Equation
PBF	Position Based Fluids
PCISPH	Predictive-Corrective Incompressible SPH
PDE	Partial Differential Equation
PPE	Pressure Poisson Equation
RAM	Random Access Memory
SDF	Signed Distance Field
SPH	Smoothed Particle Hydrodynamics
VNA	Virtual Node Algorithm
VR	Virtual Reality
XFE	eXtended Finite Element
XFEM	eXtended Finite Element Method
XSPH	(X stands for "unkown factor") Smoothed Particle Hydrodynamics

Chapter 1

Introduction

In our natural environment we interact with solid objects and fluids on a daily basis. What seems natural and intuitive to us is actually the result of complex physical phenomena, such as turbulent flow, large deformations, fracturing *etc*. In the last decades technology has evolved rapidly and applications for the purpose of mimicking nature in a digital environment are on the rise. This does not only include the generation of realistic visual images but also the physically accurate simulation of the dynamic movement of solid objects and fluids to mirror natural phenomena as closely as possible. Examples for these applications, just to mention a few, are special effect productions for feature films, fully animated movies, virtual prototyping, *Virtual Reality* (VR) applications, medical simulators, or computer games. Consequently, there is a steadily increasing demand for sophisticated techniques for the physics based simulation of solids and fluids in virtual environments.

In recent years, the computer graphics research community has greatly contributed to the development of novel techniques to simulate solids and fluids in a physically plausible way. While the vast majority of the developed methods focuses on the simulation of effects within homogeneous materials, the demand for simulations of even more complex phenomena is growing continuously. Several physical phenomena can be described by extending existing mathematical models by interface models. An interface is a two-dimensional surface embedded in three-dimensional space that acts as a separation layer. Examples for applications of interfaces as separation layers are: the surface of a fluid or a solid object, a sudden transition between different materials, or even the location and propagation of cuts, cracks, and tears within solid materials.

Consider the examples illustrated in Figure 1.1. The leftmost image depicts a hand punching into a pool of water causing splashes. In this scenario three physical entities are interacting: the solid hand exerts forces on the water, the water displaces and creates complex splashes that interact with the surrounding air. In order to mathematically model this effect, the interfaces representing the surface of the hand, and the free surfaces of water and air have to be modeled. Moreover, the water might be contained in a basin restricting the domain in which the water can move. This boundary of the basin might then also be interpreted as a physical interface separating the water from the basin's solid material. The second image in Figure 1.1 depicts a thick slice of meat. While the steak can be classified as a single solid object, it has a highly non-homogeneous structure. It is rather composed of several layers of organic material, *i.e.*, muscle tissue, fat tissue and bones, each having individual physical properties, *e.g.*, density, stiffness, *etc*. The transitions between the layers are extremely sudden and can therefore be idealized as interface surface of infinitesimal thickness. The last image (Figure 1.1, right) shows a cracked window. Although, the material can be assumed to be homogeneous, the physical behavior of the glass changes significantly due to the material failures scattered across the window. Again, providing a suitable interface description of the crack surfaces is inevitable for mathematical modeling.

Figure 1.1: Interface phenomena in nature. Left[1]: Water interacts with air and a hand. Interfaces can be used to describe the layers separating the matter. Middle[2]: T-bone steak consists of several organic materials adhering to each other. Right[3]: Cracks in a glass window separate individual shards from another.

Explicit and Implicit Interface Representations

For the numerical simulation of interface problems, suitable representations to model the geometry of an interface have to be identified. In general, there are two major strategies to describe an interface surface, namely *explicit* and *implicit* representations. Let $\Omega \subset \mathbb{R}^2$ be a domain that contains all points on the two-dimensional surface in a local frame. Then, an explicit representation implies that the interface geometry is represented as a parametric surface, *i.e.*, a function $f : \Omega \to \mathbb{R}^3$ that maps a point on the surface to a certain point in the three-dimensional world space. As it is often practically impossible to represent a complex interface resulting from real world problems using a single analytical expression, the parametric surface is frequently decomposed into a piecewise definition. In practice such a piecewise definition can, for example, be realized as a triangle mesh, where each triangle represents a local parametrized bounded planar region on the surface. In contrast, interfaces can also be represented implicitly. In this case, the interface surface is defined as the zero contour of a function $\Phi : \mathbb{R}^3 \to \mathbb{R}$, *i.e.*, $f(\Omega) = \{\mathbf{x} \mid \Phi(\mathbf{x}) = 0\}$. Again, it is practically impossible to find an analytic expression for arbitrary interface geometries. For that reason the implicit description is often discretized using a volumetric mesh or a point sampling.

There is no correct answer to the question which representation strategy is utterly superior. The best choice rather depends on the specific application and its requirements. Let us discuss two examples in which each of the two strategies is more useful. Imagine an interface is used to describe the free surface of a fluid. Due to the movement of the fluid, the surface deforms strongly and undergoes frequent topological changes. In this case the usage of an implicit description can be highly beneficial as the discretization values of the implicit function can simply be advected following physical laws to displace the zero contour. In this way topological changes can be directly captured without any additional effort. In contrast, tracking a fluid surface explicitly is a tedious task as the discrete representation has to be adapted frequently. This usually also involves numerous intersection tests and moreover requires optimization techniques to maintain a good discretization quality.

[1]"Making a splash" by RhiPh0tography is licensed under CC BY-NC-ND 2.0
[2]"Beef Loin Porterhouse Steak" by Artizone is licensed under CC BY-NC-ND 2.0
[3]"Shattered glass" by DaveBleasdale is licensed under CC BY 2.0

For the second example, consider rendering an image using a raytracer. Raytracing requires the computation of countless intersection tests between rays and an object's boundary surface. These tests are efficient and simple when the surface is represented using an explicit triangle mesh as the evaluation is equal to computing the intersection points between several planes and a line. In contrast, computing intersections between rays and implicit surfaces is complex due to the fact that each intersection test is a root finding problem and thus typically inefficient. Please note that in certain cases an explicit tracking of free fluid surfaces or an implicit surface representation for ray tracing might still be beneficial depending on the specific problem.

The pure representation of surfaces is not the only challenge simulators for physical interface problems are facing. Further challenges include the coupling with existing discretization techniques, the numerical integration over domains intersected by the interface, or the computation of fluxes transmitted through the interface. However, approaches to tackle these challenges are highly dependent on the simulation method and also very specific to certain applications. In this regard, this thesis proposes novel techniques for the simulation of interface phenomena with solids and fluids using both explicit and implicit interface representations. Furthermore, a novel technique for an efficient generation of discrete implicit interface representations is proposed.

1.1 Outline

This thesis is structured as follows. Chapter 2 describes the foundations of solid and fluid modeling on the basis of continuum mechanics. The term continuum, kinematic laws, local and global balance laws, the concept of stress, constitutive models for solids and fluids and specific boundary value problems tailored to specific problems are introduced. The remainder of the thesis is organized into three parts. The first part is dedicated to the simulation of cutting and fracture in deformable solids and covers Chapters 3 and 4. In Chapter 3, a novel method for the physically based simulation of brittle fracture using a stress analysis based on adaptive tetrahedral meshes is presented. A novel robust method to simulate cutting of deformable objects using the extended finite element method is proposed in Chapter 4. The second part – Chapter 5 – describes a novel method to discretize signed distance fields using an adaptive hexahedral grid in combination with higher-order polynomials of variable degree. The third and final part covering Chapters 6 and 7 is dedicated to the simulation of incompressible fluids using the "Smoothed Particle Hydrodynamics" formalism. Chapter 6 presents a novel method to enforce incompressibility using a solver that ensures that the density field is constant throughout the fluid and that maintains a divergence-free velocity field. In Chapter 7 an approach for handling non-penetration boundary conditions using the novel concept of implicit density maps is presented. Finally, Chapter 8 draws conclusions from the work discussed in the preceding chapters and discusses potential directions for future work.

1.2 Contributions

The contributions presented in this thesis are summarized in the following. Please note that all work has been published at refereed conferences or in peer-reviewed journals. In consideration of PromO § 5

Abs. 3 of RWTH Aachen University I would like to affirm that all listed contributions are my original work and were only supported by my colleagues and supervisors unless stated otherwise.

- *Adaptive tetrahedral meshes for brittle fracture simulation*, [**Koschier** *et al.* 2014]: A novel, reversible spatially adaptive finite element discretization for linear elasticity based on tetrahedral meshes for the simulation of brittle fracture is presented. Dynamic objects are treated as rigid bodies until they encounter a collision. Subsequently, a stress analysis is performed and the underlying mesh is successively adapted guided by a stress-based criterion in order to accurately determine highly stressed regions. The adaption scheme is designed to preserve the quality of the input mesh to a large extent. Moreover, it has the advantages that it does not alter the boundary geometry such that geometric features are maintained and that it is exclusively based on topological operations. The method is the first to perform a spatially adaptive stress analysis on tetrahedral meshes for the physically based simulation of brittle fracture in the context of computer graphics.

- *Robust cutting simulation using extended finite elements*, [**Koschier** *et al.* 2017a]: A robust remeshing-free algorithm for the simulation of cutting of deformable objects on the basis of the *eXtended Finite Element Method* (XFEM) with fully implicit time integration is presented. Using this formulation it is possible to capture a cut modeled by a discontinuity in the solution of the underlying *Partial Differential Equation* (PDE) without the requirement to perform any modifications on the discretization mesh. One of the biggest challenges for implementations of the XFEM in combination with discontinuous enrichment functions is the evaluation of integrals over discontinuous integrands on tetrahedral domains. The presented cutting algorithm includes the construction of specialized quadrature rules to approximate the integrals with high accuracy. Using the novel approach, even highly complex fine-structured cuts can be simulated in a robust manner.

- *hp-adaptive generation of discrete signed distance fields*, [**Koschier** *et al.* 2016; **Koschier** *et al.* 2017b]): An algorithm for the adaptive generation of discrete higher-order polynomial *Signed Distance Field*s (SDFs) on axis-aligned hexahedral grids is presented. Using an orthonormal polynomial basis, the functions are efficiently fit to the input signed distance function on each cell. The novel error-driven construction algorithm is globally adaptive and iteratively refines the *Signed Distance Field*s (SDFs) using either spatial subdivision (h-refinement) following an octree scheme or by cell-wise adaption of the polynomial approximation's degree (p-refinement). The results produced with this technique demonstrate that the method is able to construct more accurate *Signed Distance Field*s (SDFs) at significantly lower memory consumption compared to previous approaches.

- *Simulation of incompressible fluids using SPH*, [Bender and **Koschier** 2015; Bender and **Koschier** 2017]: A novel method for the simulation of incompressible fluids based on the semidiscretization technique *Smoothed Particle Hydrodynamics* (SPH) is presented. Using a combination of two novel implicit pressure solvers both a low volume compression as well as a divergence-free velocity field can be ensured in order to maintain incompressibility. While a compression-free fluid is essential to produce a realistic physical behavior, a divergence-free velocity field drastically reduces the number of required solver iterations and increases the stability of the simulation significantly. In this way the method can handle larger time steps than previous approaches resulting in a substantial performance gain.

Please note that this work was a collaboration and was not first-authored by myself. However, the extent of my contributions to this project was significant which can be attested by the first author. The contributions not only concern coauthoring the articles but also include the development of the established techniques and the software implementation. Moreover, I would like to affirm that the work has not been published elsewhere and has not been used in a thesis before.

- *Density Maps for improved SPH boundary handling*, [**Koschier** and Bender 2017]:

  The novel concept of density maps for robust handling of static and dynamic boundaries in fluid simulations based on SPH is presented. In contrast to the vast majority of existing approaches the presented method uses an implicit discretization for a continuous extension of the density field throughout solid boundaries. Using the novel representation the accuracy of density and density gradient evaluations in boundary regions are enhanced and the computational efficiency is improved using efficient lookups into the density maps. This strategy not only removes the necessity to sample boundary surfaces with particles, but also decouples the particle size from the number of sample points required to represent the boundary. Several comparisons show that the representation is more accurate than particle samplings, especially for smooth curved boundaries, and even outperforms one of the most recent sampling based techniques.

1.3 List of Publications

This thesis is mainly based on scientific articles that were originally published in peer-reviewed journals or at refereed conferences. A list of the relevant work can be found below.

D. Koschier, S. Lipponer, and J. Bender (2014). "Adaptive Tetrahedral Meshes for Brittle Fracture Simulation". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 1–10.

D. Koschier, C. Deul, and J. Bender (2016). "Hierarchical hp-Adaptive Signed Distance Fields". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 1–10.

D. Koschier and J. Bender (July 2017). "Density Maps for Improved SPH Boundary Handling". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 1–10.

D. Koschier, J. Bender, and N. Thuerey (2017a). "Robust eXtended Finite Elements for Complex Cutting of Deformables". *ACM Transactions on Graphics* 36.4, 55:1–55:13.

D. Koschier, C. Deul, M. Brand, and J. Bender (2017b). "An hp-Adaptive Discretization Algorithm for Signed Distance Field Generation". *IEEE Transactions on Visualization & Computer Graphics* 23.10, pp. 2208–2221.

J. Bender and **D. Koschier** (2015). "Divergence-Free Smoothed Particle Hydrodynamics". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 1–9.

J. Bender and **D. Koschier** (2017). "Divergence-Free SPH for Incompressible and Viscous Fluids". *IEEE Transactions on Visualization & Computer Graphics* 23.3, pp. 1193–1206.

Other published work that I coauthored can be found in the following list.

J. Bender, **D. Koschier**, P. Charrier, and D. Weber (2014b). "Position-Based Simulation of Continuous Materials". *Computers & Graphics* 44.1, pp. 1–10.

M. WEILER, **D. KOSCHIER**, and J. BENDER (2016). "Projective Fluids". *ACM SIGGRAPH Motion in Games*, pp. 1–6.

J. BENDER, **D. KOSCHIER**, T. KUGELSTADT, and M. WEILER (July 2017). "A Micropolar Material Model for Turbulent SPH Fluids". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 1–8.

M. WEILER, **D. KOSCHIER**, M. BRAND, and J. BENDER (2018). "A Physically Consistent Implicit Viscosity Solver for SPH Fluids". *Computer Graphics Forum*.

Chapter 2

Foundations of Solid and Fluid Simulation

Physics teaches us that all the matter surrounding us and even the matter we consist of is formed out of molecules that in turn are composed of atoms. Each of these incredibly small particles contributes a certain mass to the portion of matter it belongs to. Therefore, we know that the distribution of mass within matter is not continuous but can rather be interpreted as a system of discrete mass points. Despite this fact, many macroscopic mechanical phenomena can be well-described when the corresponding matter is idealized as a *continuum*, *i.e.,* a region of continuously distributed mass. Most of these phenomena are subject to the interaction among solid bodies, liquids or gases exhibited to external influences, *e.g.,* temperature, external loading, or gravity. This idealization then implies that a portion of matter can always be divided into smaller portions independent of the size of the regions. This in turn, confirms the theoretical existence of a *material particle*, *i.e.,* a portion of matter contained in an infinitesimal volume, within this model. The *continuum theory* then aims to model macroscopic physical phenomena and neglects effects that can be observed on microscales.

This chapter gives a brief introduction into the field of continuum mechanics for the mathematical modeling of solids and fluids and is based on the works of LAI *et al.* [2009], ABEYARATNE [2012], and TSAKMAKIS [2013].

2.1 Continuum Mechanics

In the field of continuum mechanics the behavior of bodies with continuously distributed mass exhibited to external loads is studied. The materials that are most frequently modeled using continuum theory are elastic and plastic solids and Newtonian fluids. Based on mechanical axioms resulting from experimental physics or simple observations, physical principles can be deduced to derive suitable mathematical models. Consequently, solid or fluid matter in a continuous setting is characterized by its kinematics, balance laws, constitutive laws and the second law of thermodynamics. The kinematics describe the motion and deformation of the continuum, usually using differential measures. In case of an elastic solid body this expresses the relation between the spatial displacement of each material particle and the resulting strain at the particle location. The balance laws cover, among others, the conservation of linear momentum corresponding to Newton's second law and the conservation of angular momentum. Constitutive laws model the relation between certain field quantities and mechanical stress. Examples for these field quantities are strain, strain-rate, or temperature. The corresponding equations therefore model the inherent behavior of a material. Consider the following examples. While solid bodies being subject to small deformations can be described by linear constitutive models, solids undergoing large deformations require more elaborate non-linear constitutive equations. Also fluids follow the exact same balance laws as elastic solids but mainly differ in the constitutive model as they are subject to purely plastic deformation and therefore never return to their initial state. Finally, the second law of thermodynamics is usually evaluated for a given constitutive law in order to determine if a material is thermodynamically valid. Physically interpreted the law states that the entropy of an isolated system never decreases. From that it follows that the heat in a closed system never flows from a colder to a hotter region. Another implication is that a closed system can never produce but only conserve or dissipate energy. In solid and fluid mechanics the law is usually expressed in form of the Clausius-Duhem inequality to test the validity of constitutive relations.

Continuum

A three-dimensional *body* $\mathcal{B}$ is defined by a set of material particles $\mathfrak{X} \in \mathcal{B}$. The concept of a body and material particles is inherently abstract such that $\mathcal{B}$ is intended to represent a portion of matter as occurring in nature. In order to give these abstract entities a geometric meaning the concept of configurations has to be introduced. Mathematically, a configuration $\mathcal{X}$ is defined as a one-to-one mapping

$$\mathcal{X} : \mathcal{B} \to \Omega \subset \mathbb{E}^3, \quad \mathbf{x} = \mathcal{X}(\mathfrak{X}).$$

that maps a material particle $\mathfrak{X}$ to its definite location $\mathbf{x}$ in the domain Ω occupied by the body in three-dimensional Euclidean space $\mathbb{E}^3$. Please note, that the inverse mapping $\mathcal{X}^{-1} : \Omega \to \mathcal{B}$ also exists since $\mathcal{X}$ describes one-to-one relations.

In order to model explicit problems using continuum theory it is beneficial to define one unique configuration as reference configuration. For elastic materials this can be (but does not have to be) the configuration in which the objects rests, *i.e.,* the body is not subject to any internal stresses. In case of liquids or gases it is convenient to choose this configuration as the initial position of the fluid. In the following this configuration will be referred to as the *reference configuration*

$$\mathcal{X}_R : \mathcal{B} \to \Omega_R, \quad \boldsymbol{\xi} = \mathcal{X}_R(\mathfrak{X})$$

that maps a material particle $\mathfrak{X}$ to a position $\boldsymbol{\xi}$ in the reference domain Ω_R. The definition of such a reference configuration is especially important for modeling elastic solids. In this case the internal forces are directly dependent on the deviation of a deformed configuration relative to the undeformed configuration, *i.e.,* the reference configuration. Although less significant, the definition of a reference configuration can also be very helpful when fluids are modeled. The configuration can give the investigator insight concerning the initial volume of the fluid, *e.g.,* when incompressibility or other geometric characteristics are considered. Given an arbitrary configuration $\mathcal{X}$ of the body $\mathcal{B}$ the deformation from the reference configuration $\mathcal{X}_R$ can be measured using the deformation mapping $\mathfrak{D} = \mathcal{X} \circ \mathcal{X}_R^{-1}$. Here the $\circ$-operator denotes a composition of functions such that $(f \circ g)(x) = f(g(x))$. Please see Figure 2.1 for a graphical illustration of the configurations and the mappings between them.

2.2 Motion and Kinematics

A *motion* describes the transition from one configuration to another on a certain path. This transition can be parameterized using a single parameter. Physically, this parameterization can be used to represent the deformation of a continuum with respect to its reference configuration over time t within

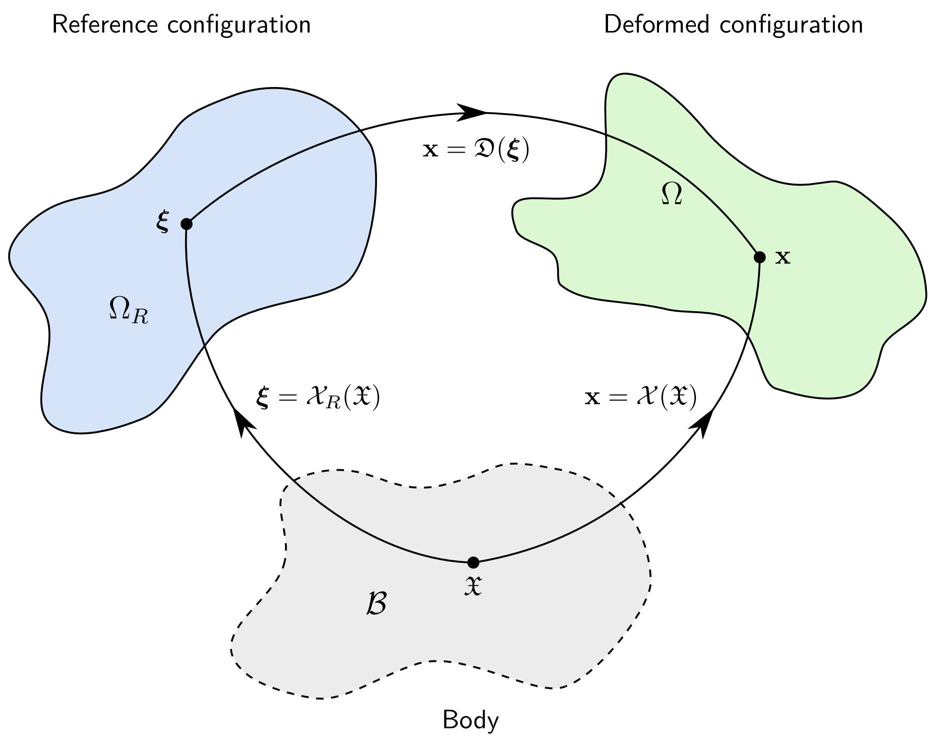

Reference configuration
Deformed configuration
$\mathbf{x} = \mathfrak{D}(\boldsymbol{\xi})$
$\boldsymbol{\xi}$
Ω
$\mathbf{x}$
Ω_R
$\boldsymbol{\xi} = \mathcal{X}_R(\mathfrak{X})$
$\mathbf{x} = \mathcal{X}(\mathfrak{X})$
$\mathcal{B}$
$\mathfrak{X}$
Body

Figure 2.1: Configurations and deformation mapping.

a time interval $[t_0, t_1]$. This finally results in a family of configurations $\mathcal{X}(\mathfrak{X}, t) : \mathcal{B} \to \Omega_t$ where Ω_t denotes the spatial domain occupied by the body $\mathcal{B}$ at time t. From here onwards, the following abbreviations will be used to ensure readability: $\mathcal{X}_t(\mathfrak{X}) = \mathcal{X}(\mathfrak{X}, t)$, $\mathcal{X}_i(\mathfrak{X}) = \mathcal{X}(\mathfrak{X}, t_i)$. Also $\mathcal{X}_0$ will be denoted as the *initial configuration* in the remainder of this thesis. Please note that the reference configuration is often chosen as the initial configuration for the sake of convenience, *i.e.*, $\mathcal{X}_R \equiv \mathcal{X}_0$, $\Omega_R \equiv \Omega_0$.

In the course of the motion each material particle follows a trajectory described by the curve $\mathbf{x}_t$. In general, there are four types of coordinates in which the given trajectory can be described: in material coordinates, reference coordinates, in Lagrangian coordinates or Eulerian coordinates. This is also true for the values that a time-dependent function α_t defined on $\mathcal{B}$ can take. The different descriptions can then be represented as follows:

- Material description:
$$\alpha_t = \alpha_t^M(\mathfrak{X}), \quad \mathbf{x}_t = \mathbf{x}_t^M(\mathfrak{X})$$
- Reference description:
$$\alpha_t = \alpha_t^R(\boldsymbol{\xi}), \quad \mathbf{x}_t = \mathbf{x}_t^R(\boldsymbol{\xi})$$
- Lagrangian description:
$$\alpha_t = \alpha_t^L(\mathbf{x}_0), \quad \mathbf{x}_t = \mathbf{x}_t^L(\mathbf{x}_0)$$
- Eulerian description:
$$\alpha_t = \alpha_t^E(\mathbf{x}_t), \quad \mathbf{x}_t = \mathbf{x}_t$$

In the Eulerian description no extra mapping for the vector $\mathbf{x}_t$ is required, because the trajectory is measured in the current/deformed configuration. It is also worth noting that the reference and the Lagrangian description are equivalent if the reference configuration is chosen as the initial configuration, *i.e.,* $\boldsymbol{\xi} \equiv \mathbf{x}_0$ if $\mathcal{X}_R \equiv \mathcal{X}_0$. In the following the index t will be left out for the sake of brevity. This means that if a time-dependent quantity is index-free, it represents the given quantity at time t.

Besides describing the trajectory $\mathbf{x}$ of a material particle $\mathfrak{X}$, it is often necessary to calculate particle velocity $\mathbf{v}$ and acceleration $\mathbf{a}$ for modeling, analysis, or numerical solving. These vector fields are defined as time derivatives of the particle trajectory, *i.e.,* $\mathbf{v} := \dot{\mathbf{x}}$, $\mathbf{a} := \ddot{\mathbf{x}}$. However, for physical quantities represented by scalar or tensor fields on the body $\mathcal{B}$, the (partial) time derivative has different meanings when described in Lagrangian or Eulerian coordinates. In order to express the absolute time rate of change of a given quantity the *material derivative* $D(\cdot)/Dt$ is introduced. In the following, the material derivative of a quantity will alternatively be denoted with an overdot for brevity. Hence, the time rate of change of a tensor field α in Lagrangian and Eulerian coordinates are defined as

$$\text{(Lagrangian)} \quad \dot{\boldsymbol{\alpha}} = \frac{D\boldsymbol{\alpha}}{Dt} = \frac{\partial \boldsymbol{\alpha}^L}{\partial t} \quad \text{and} \quad \dot{\boldsymbol{\alpha}} = \frac{D\boldsymbol{\alpha}}{Dt} = \frac{\partial \boldsymbol{\alpha}^E}{\partial t} + \underbrace{(\mathbf{v} \cdot \nabla_{\mathbf{x}})\, \boldsymbol{\alpha}^E}_{\substack{\text{convective} \\ \text{derivative}}}. \quad \text{(Eulerian)}$$

While the partial derivative of α in Lagrangian coordinates coincides with the material derivative the Eulerian field has an additional term that is referred to as convective derivative.

Deformation

When a solid body is modeled using the presented continuum theory, one has to introduce mathematical tools to geometrically measure the body's deformation and strain. The definition of suitable measures will be important to model constitutive laws for elasticity relating the geometric strain with mechanical stress. As motivation one can consider a continuum represented in reference configuration and an arbitrary deformed configuration (*cf.* Figure 2.2). Now a parametrized curve $\mathbf{C}(\lambda)$ within the continuum connecting a "string" of material particles can be chosen. The material curve naturally takes different shapes in each of the considered configurations, *i.e.,* $\mathbf{C}_R = \mathcal{X}_R(\mathbf{C})$ and $\mathbf{C}_t = \mathcal{X}_t(\mathbf{C})$. In order to measure the deformation of points on the line, a relation between the infinitesimal line elements $d\mathbf{C}_R, d\mathbf{C}_t$ in reference and deformed configuration has to be found. This can be achieved by simple derivation and application of the chain rule:

$$\frac{\partial \mathbf{C}_t}{\partial \lambda} = \frac{\partial \mathfrak{D}(\mathbf{C}_R)}{\partial \lambda} = \nabla_{\boldsymbol{\xi}} \mathfrak{D}\, \frac{\partial \mathbf{C}_R}{\partial \lambda}.$$
$$\Rightarrow d\mathbf{C}_t = \nabla_{\boldsymbol{\xi}} \mathfrak{D}\; d\mathbf{C}_R.$$

This yields a linear relation between line elements on the curve among the reference and any deformed configuration. The two-point tensor relating the quantities is the gradient of the deformation mapping with respect to the reference coordinates. This gradient also directly follows from a Taylor series expansion in $\boldsymbol{\xi}$ ($t = \text{const}$) of first order of the deformation mapping at a position $\boldsymbol{\xi}_0 \in \Omega_R$:

$$\mathfrak{D}(\boldsymbol{\xi}_0) = \mathbf{x}_0 + \nabla_{\boldsymbol{\xi}} \mathfrak{D}|_{\boldsymbol{\xi}_0} (\boldsymbol{\xi} - \boldsymbol{\xi}_0) + \mathbf{r}(\boldsymbol{\xi}_0, \boldsymbol{\xi} - \boldsymbol{\xi}_0), \quad \lim_{\|\boldsymbol{\xi} - \boldsymbol{\xi}_0\| \to 0} \frac{\mathbf{r}(\boldsymbol{\xi}_0, \boldsymbol{\xi} - \boldsymbol{\xi}_0)}{\|\boldsymbol{\xi} - \boldsymbol{\xi}_0\|} = \mathbf{0},$$

where $\mathbf{r}$ denotes the remainder of the Taylor series.

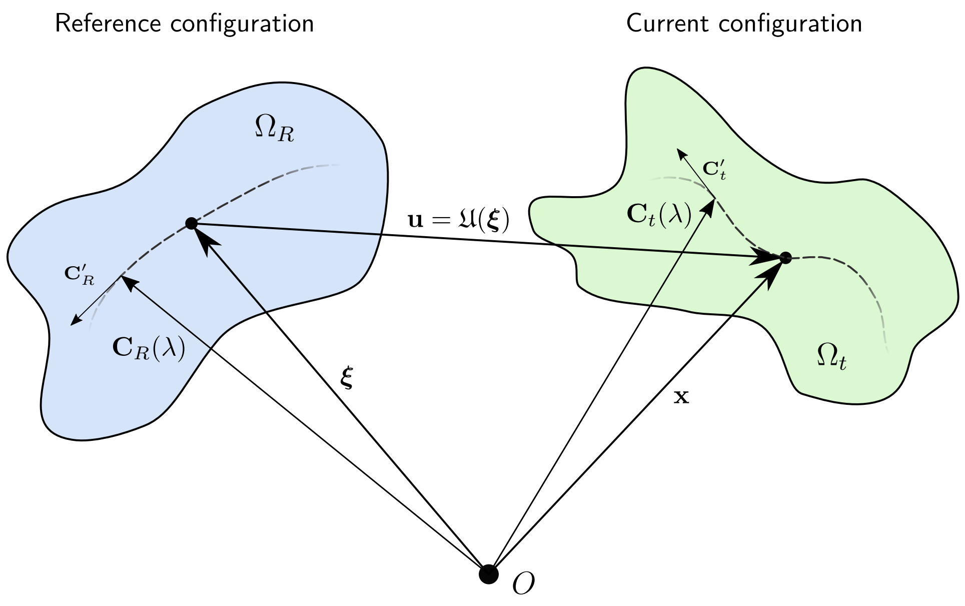


Figure 2.2: The displacement mapping and deformation of a material path. O denotes the origin of a Cartesian coordinate system in Euclidean space.

Deformation Gradient

In the following, this gradient will be referred to as *deformation gradient* and will be denoted with $\mathbf{F} = \nabla_{\boldsymbol{\xi}}\mathfrak{D}$. From the previous insight that $\mathbf{F}$ transforms line elements from reference into deformed space we can also deduce that

$$d\mathbf{x} = \mathbf{F}\, d\boldsymbol{\xi}. \tag{2.1}$$

One of the most important properties of the deformation gradient $\mathbf{F}$ is that it can be factorized into a proper orthogonal tensor and a symmetric tensor as long as $\mathbf{F}$ does not represent a reflection. This decomposition is referred to as *polar decomposition*. Presuming that $\det \mathbf{F} > 0$, the decomposition is mathematically defined as

$$\mathbf{F} = \mathbf{R}\mathbf{U} = \mathbf{V}\mathbf{R}, \quad \text{with} \quad \mathbf{R} = \mathbf{R}^{-T}, \mathbf{U} = \mathbf{U}^T, \mathbf{V} = \mathbf{V}^T. \tag{2.2}$$

The symmetric and positive definite tensors $\mathbf{U}$ and $\mathbf{V}$ are often referred to as *right* and *left stretch tensor*. Also existence and uniqueness of the polar decomposition under the given requirement can be proved. For further details on the proof I would like to refer the reader to the book of LAI *et al.* [2009]. Finally, the second-order tensor $\mathbf{F}$ can be used as a measure for (point-wise) deformation.

Displacement and Displacement Gradient

The displacement mapping $\mathfrak{U} : \Omega_R \to \mathbb{R}^3$ maps the position of a material particle in reference configuration $\mathcal{X}_R$ to its displacement vector $\mathbf{u}_t$ at time t in deformed configuration $\mathfrak{X}$, *i.e.*, $\mathbf{u}_t = \mathfrak{U}_t(\boldsymbol{\xi})$. Again, the index t denoting the time point associated with the deformed configuration will be dropped in the remainder to improve readability. Analogously to the deformation gradient, the deformation of a line element can also be expressed using the displacement gradient $\mathbf{H} := \frac{\partial \mathfrak{U}}{\partial \boldsymbol{\xi}}$ resulting in $d\mathbf{x} = (\mathbf{H} + \mathbb{1})d\boldsymbol{\xi}$.

Consequently, the deformation gradient and the displacement gradient are related via the identity $\mathbf{F} = \mathbf{H} + \mathbb{1}$, where $\mathbb{1}$ denotes the identity matrix.

Strain

The main question motivating a strain measure is: How can one quantify the deformation of a material particle exclusive a potential rigid body motion in the near-field of the particle? Unfortunately, the deformation gradient $\mathbf{F}$ is not suitable for measuring strain as the tensor contains rigid body rotations. To construct suitable representations for strain measures one has to take care that rigid body transformations are eliminated. In general, the strain tensors can be classified into Lagrangian and Eulerian measures. Analogously to the corresponding material descriptions, Lagrangian strain tensors measure strain in the reference configuration while Eulerian strain tensors represent the quantity in deformed space. In the following, several strain measures will be specified. The focus will, however, lie on Lagrangian measures as they are more relevant for this thesis.

Green-Lagrange Strain Tensor

A starting point to describe rigid body transformation free deformation of a line element is to measure its (quadratic) length difference Δ

$$\Delta := \frac{1}{2}\left(\|d\mathbf{x}\|^2 - \|d\boldsymbol{\xi}\|^2\right) \tag{2.3}$$

as its length is not influenced by rigid body motions. Plugging Equation (2.1) into Equation (2.3) yields

$$\begin{aligned}\Delta &= \frac{1}{2}(\mathbf{F}d\boldsymbol{\xi} \cdot \mathbf{F}d\boldsymbol{\xi} - d\boldsymbol{\xi} \cdot d\boldsymbol{\xi}) \\ &= \frac{1}{2}\left(d\boldsymbol{\xi} \cdot \mathbf{F}^T\mathbf{F}d\boldsymbol{\xi} - d\boldsymbol{\xi} \cdot d\boldsymbol{\xi}\right) \\ &= d\boldsymbol{\xi} \cdot \underbrace{\left[\frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbb{1}\right)\right]}_{\mathbf{E}} d\boldsymbol{\xi},\end{aligned}$$

where $\mathbf{E}$ is referred to as the *Green-Lagrange strain tensor* or short *Green strain tensor*. The independence of this strain measure from rigid body rotations also becomes intuitively clear when considering that $\mathbf{F}$ can be decomposed using the polar decomposition (*cf.* Equation (2.2)) yielding

$$\begin{aligned}\mathbf{E} &= \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbb{1}\right) \\ &= \frac{1}{2}\left(\mathbf{U}^T\mathbf{R}^T\mathbf{R}\mathbf{U} - \mathbb{1}\right) \\ &= \frac{1}{2}\left(\mathbf{U}^2 - \mathbb{1}\right).\end{aligned}$$

In this way, one can see that the rigid body rotation contained in $\mathbf{F}$ is eliminated. It is also worth noting that the Green strain tensor is a Lagrangian strain measure since it can also be interpreted as a function mapping line elements from reference to reference space, *i.e.*, $\mathbf{E} : \mathcal{T}_{\boldsymbol{\xi}}\,\Omega_R \times \mathcal{T}_{\boldsymbol{\xi}}\,\Omega_R \to \mathbb{R}$, where $\mathcal{T}_{\boldsymbol{\xi}}\,\Omega_R$ denotes the tangent space of Ω_R at point $\boldsymbol{\xi}$. There is also an equivalent Eulerian strain measure $\mathbf{A} := \frac{1}{2}\left(\mathbb{1} - \mathbf{F}^{T^{-1}}\mathbf{F}^{-1}\right)$ which is referred to as the *Almansi strain tensor* such that $\mathbf{A} : \mathcal{T}_{\mathbf{x}}\,\Omega \times \mathcal{T}_{\mathbf{x}}\,\Omega \to \mathbb{R}$. The Lagrangian and the Eulerian measure are then related via the identity $\mathbf{E} = \mathbf{F}^T\mathbf{A}\mathbf{F}$.

Cauchy/Linearized Strain Tensor

The previously introduced strain measures ensure that no rigid body rotations are evident. However, the measures are non-linear (or more specifically quadratic) in the material particle position. If a deformable object is subject to very small deformations a linearized strain measure can be sufficient to describe the object's behavior adequately. This in turn simplifies analytical or numerical analyses as non-linearity is avoided. The linearization can then be realized by developing the Green strain into a Taylor series of first order

$$\begin{aligned}\mathbf{E} &= \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbb{1}\right)\\ &= \frac{1}{2}\left((\mathbf{H}+\mathbb{1})^T(\mathbf{H}+\mathbb{1}) - \mathbb{1}\right)\\ &= \frac{1}{2}\left(\mathbf{H}+\mathbf{H}^T\right) + \frac{1}{2}\mathbf{H}^T\mathbf{H}\\ &\approx \frac{1}{2}\left(\mathbf{H}+\mathbf{H}^T\right) =: \boldsymbol{\varepsilon},\end{aligned}$$

where $\boldsymbol{\varepsilon}$ is referred to as either *Cauchy strain tensor* or *linearized strain tensor*. It should, however, be well-understood that this measure <u>is</u> in fact influenced by rigid body rotations due to the linearization.

Biot Strain Tensor

As previously mentioned, a linear strain measure is beneficial when it comes to numerical computations because less non-linearity means better convergence when solving the arising equation systems. However, for deformable objects subject to large deformations a rotationally invariant strain measure is still necessary. In order to keep the strain tensor "as linear as possible" the right stretch tensor can be directly used, *i.e.,*

$$\mathbf{E}_{\text{Biot}} = \mathbf{U} - \mathbb{1}.$$

The Lagrangian measure is rotationally invariant by definition as $\mathbf{U}$ represents the symmetric part of the deformation gradient's polar decomposition and is referred to as *Biot strain tensor*.

Velocity Gradient and Strain Rate

In some cases it is not enough to solely quantify deformation as source for mechanical stresses. Especially, for viscous effects in solids or fluids the rate of deformation is also important. The deformation rate can then simply be defined as the time derivative of deformation or displacement gradient, *i.e.,*

$$\dot{\mathbf{F}} = \dot{\mathbf{H}} = \frac{d}{dt}\frac{\partial \mathfrak{U}}{\partial \boldsymbol{\xi}} = \nabla \mathbf{v}.$$

From this quantity strain rate measures can be derived. One example is then the linearized strain rate tensor

$$\dot{\boldsymbol{\epsilon}} = \frac{1}{2}(\dot{\mathbf{H}} + \dot{\mathbf{H}}^T) = \frac{1}{2}(\nabla\mathbf{v} + \nabla\mathbf{v}^T).$$

However, not relevant for this thesis, also more complex non-linear strain rate measures can be introduced following the same idea. For further details, I would like to refer the reader to the work of BONET and WOOD [1997].

2.3 Global and Local Mechanical Balance Laws

In this section, the concept of global and local balance laws will be introduced. Based on this concept, the mechanical principles of mass conservation and linear and angular momentum conservation will be derived.

Consider a body $\mathcal{B}$ subject to a motion $\mathcal{X} : \mathcal{B} \to \Omega$ in the time interval $t \in [t_0, t_1]$. Let $\mathcal{P} \subset \mathcal{B}$ be a region that is part of the body $\mathcal{B}$ and $\Omega^{\mathcal{P}} = \mathcal{X}(\mathcal{P})$ be the domain in the deformed configuration at time t. Here, it should be pointed out that $\mathcal{P}$ is composed of a fixed set of material particles $\mathfrak{X} \in \mathcal{P}$. This implies that independent of the deforming motion the set of particle covered by the domain $\Omega^{\mathcal{P}}$ describes the deformed domain covering the same set of material particles. Let A further be a physical property associated with $\mathcal{P}$, then one can assume that there is a (smooth) density function $\alpha : \mathcal{B} \times \mathcal{T} \to \mathbb{R}$, where $\mathcal{T} = [t_0, t_1]$ denotes the considered time domain with start and end time t_0 and t_1, such that

$$A = \iiint\limits_{\Omega^{\mathcal{P}}} \alpha(\mathbf{x}) d\mathbf{x}. \tag{2.4}$$

When taking the total time derivative of Equation (2.4) and applying *Reynold's transport theorem* this results in the following identities

$$\begin{aligned} \dot{A} = \frac{d}{dt} \iiint\limits_{\Omega^{\mathcal{P}}} \alpha d\mathbf{x} &= \iiint\limits_{\Omega^{\mathcal{P}}} \dot{\alpha} + [\alpha \, \nabla \cdot \mathbf{v}] \, d\mathbf{x} \\ &= \iiint\limits_{\Omega^{\mathcal{P}}} \underbrace{\frac{\partial \alpha}{\partial t}}_{\substack{\text{volumetric} \\ \text{generation}}} d\mathbf{x} + \oint\!\!\!\oint\limits_{\partial \Omega^{\mathcal{P}}} \underbrace{(\mathbf{v} \cdot \mathbf{n}) \, \alpha \, dS}_{\text{flux}}, \end{aligned} \tag{2.5}$$

where $\mathbf{n}$ denotes the surface normal of the material portion's boundary $\partial\Omega^{\mathcal{P}}$. This model formulation is called a *balance law* as it simply states that the increase rate of the amount of α in $\mathcal{P}$ is equal to (or "balanced" with) the volumetric generation of α and the flux of α across $\partial\mathcal{P}$. It is further called *global* balance law as it only holds for the considered domain (or parts of the domain) but not necessarily for a single particle.

Based on global laws, *local balance laws*, also often called *field equations*, and boundary conditions can be derived. This derivation is usually done by localization. While not explicitly formulated here the outline of the localization process is as follows. The boundary integral over the flux in Equation (2.5) is transformed into a volume integral using the divergence theorem. Since it is presumed that all integrands are smooth and since the integration domain $\Omega^{\mathcal{P}}$ is a subset of Ω, the *localization theorem* can be applied and the balance law must also hold point-wise. In the following, several field equations will be derived. These are namely the mass conservation law, the law of linear momentum conservation and the law of angular momentum conservation.

Since all numerical simulation methods presented in this thesis are based on discretizations in Lagrangian coordinates, the balance laws will further be formulated with respect to a reference configuration $\mathcal{X}_R$.

Mass Conservation Law

A set of material particles $\mathcal{P} \subset \mathcal{B}$ that is part of a body $\mathcal{B}$ is associated with a certain mass m. Mathematically this can be represented by an integral over the material subset of a scalar field $\rho : \mathcal{B} \times \mathcal{T} \to \mathbb{R}^+$ that maps a material particle $\mathfrak{X}$ at time t to its mass density $\hat{\rho} = \rho(\mathfrak{X}, t)$. In order to improve readability, the mass density mapping $\rho(\mathfrak{X}, t)$, the mass density $\hat{\rho}$, and the mass density function that maps a position in deformed space to the material density $\bar{\rho} : \Omega \times \mathcal{T} \to \mathbb{R}^+$ will be denoted with ρ in the remainder of this work. Then, the mass m associated with a subset of material particles of a body $\mathcal{B}$ is defined as

$$m := \iiint_{\mathcal{P}} \rho(\mathfrak{X}, t) d\mathfrak{X} = \iiint_{\Omega^{\mathcal{P}}} \rho(\mathbf{x}) d\mathbf{x}. \tag{2.6}$$

As this work will consider only closed systems without mass flux, a global conservation law can be derived by differentiating Equation (2.6) with respect to the time t and by applying Reynolds transport theorem. This yields

$$\begin{aligned} \dot{m} &= \frac{d}{dt} \iiint_{\Omega^{\mathcal{P}}} \rho d\mathbf{x} \\ \text{Reynold's theorem} \rightarrow &= \iiint_{\Omega^{\mathcal{P}}} \frac{\partial \rho}{\partial t} d\mathbf{x} + \oiint_{\partial \Omega^{\mathcal{P}}} (\mathbf{v} \cdot \mathbf{n})\, \rho \, d\mathbf{x} \\ \text{Divergence theorem} \rightarrow &= \iiint_{\Omega^{\mathcal{P}}} \left[\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) \right] d\mathbf{x} \overset{!}{=} 0 \end{aligned} \tag{2.7}$$

Since ρ is assumed to be smooth on $\Omega^{\mathcal{P}}$, the integral vanishes if and only if the integrand vanishes, *i.e.*,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0. \tag{2.8}$$

Equation (2.8) is usually referred to as the *continuity equation* and holds for all points in the domain Ω described by the deformed configuration $\mathcal{X}$. The transformation steps for the localization are all reversible. Therefore, the field equation (2.8) and the global mass conservation law (2.7) are equivalent. By a few further mathematical transformations the following identity can be derived.

$$\begin{gathered} \frac{\partial \rho}{\partial t} + \nabla \rho \cdot \mathbf{v} = -\rho \nabla \cdot \mathbf{v} \\ \Leftrightarrow \\ \frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{v} \end{gathered} \tag{2.9}$$

This also implies that when incompressible materials are simulated, *i.e.*, $D\rho/Dt \overset{!}{=} 0$, the velocity field must be divergence-free, *i.e.*, $\nabla \cdot \mathbf{v} = 0$ since $\rho > 0$. This is an important observation, especially when it comes to the simulation of incompressible fluids, as this means that a divergence-free velocity field has to be enforced at all times.

Balance Law of Linear Momentum

In this section the law of linear momentum conservation will be derived. Therefore, the concept of forces will be introduced first. Secondly, a mathematical definition of the linear momentum of a material portion will be defined. Finally, the global and local balance law will be derived by evaluating Newton's second and third law.

Consider a body $\mathcal{B}$ being subject to forces. The forces acting on the body will be further distinguished into *external* and *internal forces*. *External* forces represent all forces that the environment exerts on the body. This includes external body forces acting on volume portions of the body, *e.g.*, gravitational forces, and traction forces acting on surface areas of the body's boundary $\partial\mathcal{B}$, *e.g.*, contact forces. *Internal forces* are characterized by interactions of the material particles inside the body. These can, however, be reactions to external influences such as deformation, temperature changes, *etc*. In this work only internal traction forces that model the interaction with inner material regions will be considered.

In order to mathematically characterize the total force $\mathbf{F}$ acting on a material portion $\mathcal{P} \subset \mathcal{B}$ at time t given the current configuration $\mathcal{X}$ and corresponding domain $\Omega^{\mathcal{P}} = \mathcal{X}(\mathcal{P})$, one can assume that it is the result of an integral over $\Omega_{\mathcal{P}}$ and its boundary $\partial\Omega^{\mathcal{P}}$ of smooth density functions $\mathbf{b} : \Omega^{\mathcal{P}} \times \mathcal{T} \to \mathbb{R}^3$ and $\mathbf{t} : \partial\Omega^{\mathcal{P}} \times \mathcal{S} \times \mathcal{T} \to \mathbb{R}^3$, *i.e.*,

$$\mathbf{F} = \iiint_{\Omega^{\mathcal{P}}} \mathbf{b}(\mathbf{x}) d\mathbf{x} + \oiint_{\partial\Omega^{\mathcal{P}}} \mathbf{t}(\mathbf{x}, \mathbf{n}) dS,$$

where $\mathcal{S} := \{\mathbf{x} \in \mathbb{R}^3 \mid \mathbf{x} \cdot \mathbf{x} = 1\}$ denotes the group of unit vectors.

Physically, the functions $\mathbf{b}(\mathbf{x}, t)$ and $\mathbf{t}(\mathbf{x}, t)$ can be interpreted as a specific body force field and a specific traction force field representing a force per unit volume and a force per unit surface area, respectively. $\mathbf{t}$ is also often referred to as *traction* or *stress vector*. The assumption that $\mathbf{t}$ is existent not only on $\partial\Omega$ but also on $\partial\Omega^{\mathcal{P}}$, which lies in the interior of Ω, can be motivated as follows. As previously stated, it is assumed that the body is subject to external surface and body forces. The effect of these forces is then propagated throughout the body and counteracted by internal forces. This means that the forces are transported across the interface $\partial\Omega^{\mathcal{P}}$ between $\Omega^{\mathcal{P}}$ and $\Omega \setminus \Omega^{\mathcal{P}}$. The traction on an area segment ΔS on the interface is then equal to the transported force $\Delta\mathbf{F}$ acting on the surface area element ΔS with corresponding surface normal $\mathbf{n}$ (*cf.* Figure 2.3), *i.e.*,

$$\mathbf{t} = \lim_{\Delta S \to 0} \frac{\Delta\mathbf{F}}{\Delta S} = \frac{d\mathbf{F}}{dS}.$$

This model is referred to as the *Cauchy hypothesis*.

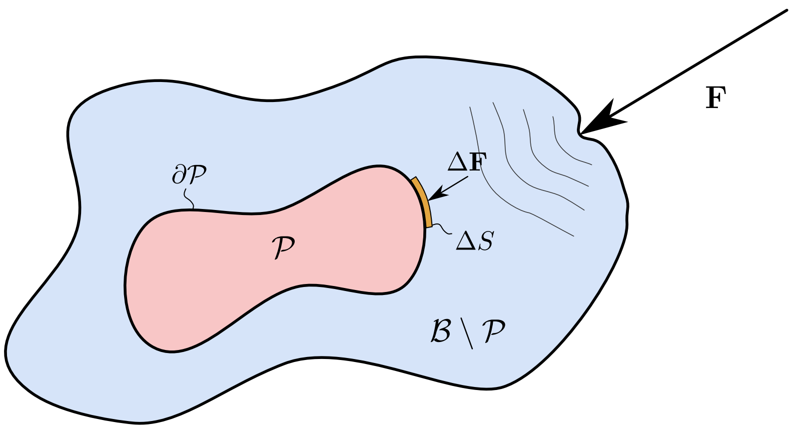


Figure 2.3: Cauchy hypothesis. A force $\mathbf{F}$ acts on the boundary of the body. The solid object reacts to the force and propagates a force field throughout the domain until it reaches $\mathcal{P}$.

Besides introducing the concept of force, the linear momentum $\mathbf{I}$ of a material portion $\mathcal{P}$ has to be defined. Again, it can be assumed that the global quantity $\mathbf{I}$ is the result of an integral over the material portion or, alternatively, its volume in the current configuration. Starting from the definition of classical mechanics, momentum is defined as product of mass and velocity. From that definition, the integral formulation of momentum can be defined as

$$\mathbf{I} := \iiint\limits_{\mathcal{P}} \rho\mathbf{v}d\mathfrak{X} = \iiint\limits_{\Omega^{\mathcal{P}}} \rho\mathbf{v}d\mathbf{x}.$$

When incorporating the mass conservation law given in Equation (2.8) the time derivative of the total momentum simplifies to

$$\begin{aligned}
\dot{\mathbf{I}} &= \frac{d}{dt}\iiint\limits_{\Omega^{\mathcal{P}}} \rho\mathbf{v}d\mathbf{x} \\
\text{Reynold's theorem} \rightarrow &= \iiint\limits_{\Omega^{\mathcal{P}}} \left[\frac{d(\rho\mathbf{v})}{dt} + (\rho\mathbf{v})(\nabla\cdot\mathbf{v})\right]d\mathbf{x} \\
&= \iiint\limits_{\Omega^{\mathcal{P}}} [\rho\dot{\mathbf{v}} + \underbrace{(\dot{\rho} + \rho\,\nabla\cdot\mathbf{v})}_{\substack{=0 \\ (cf.\ \text{Equation (2.8)})}}\mathbf{v}]d\mathbf{x} \\
&= \iiint\limits_{\Omega^{\mathcal{P}}} \rho\dot{\mathbf{v}}d\mathbf{x}.
\end{aligned} \tag{2.10}$$

Then, the global balance law for the conservation of linear momentum directly follows from *Newton's second law*. This expresses that the total momentum in a system should be in equilibrium with the total external force, such that one can set the total derivative of the total momentum $\mathbf{I}$ equal to the external force vector which yields

$$\begin{gathered}
\dot{\mathbf{I}} = \mathbf{F} \\
\Leftrightarrow \\
\iiint\limits_{\Omega^{\mathcal{P}}} \rho\dot{\mathbf{v}}d\mathbf{x} = \iiint\limits_{\Omega^{\mathcal{P}}} \mathbf{b}d\mathbf{x} + \oiint\limits_{\partial\Omega^{\mathcal{P}}} \mathbf{t}dS.
\end{gathered} \tag{2.11}$$

Newton's second law results from observations in nature and can therefore considered to be an axiom. It is also interesting to observe that the global law has the same structure as the global model balance law in Equation (2.5).

To derive a local balance law from this global formulation the boundary traction term has to be transformed into a volume integral, first. *Cauchy's stress theorem* states that there is a linear relation between the traction vector and the corresponding surface normal, *i.e.*, $\mathbf{t} = \boldsymbol{\sigma}^T\mathbf{n}$, where $\boldsymbol{\sigma}$ is a second-order tensor and referred to as the *Cauchy stress tensor*. A mathematical proof for the correctness of this theorem and the existence of the stress tensor can be found in the standard literature on continuum mechanics (see *e.g.*, [GONZALEZ and STUART 2008]). Consequently, Equation (2.11) can be rewritten to obtain the corresponding field equation

$$\iiint\limits_{\Omega^{\mathcal{P}}} \rho\dot{\mathbf{v}}d\mathbf{x} = \iiint\limits_{\Omega^{\mathcal{P}}} \mathbf{b}d\mathbf{x} + \oiint\limits_{\partial\Omega^{\mathcal{P}}} \boldsymbol{\sigma}^T\mathbf{n}dS = \iiint\limits_{\Omega^{\mathcal{P}}} \left[\nabla\cdot\boldsymbol{\sigma}^T + \mathbf{b}\right] d\mathbf{x}$$

$$\Leftrightarrow$$

$$\rho\dot{\mathbf{v}} = \nabla\cdot\boldsymbol{\sigma}^T + \mathbf{b}. \tag{2.12}$$

Finally, Equation (2.12) represents a local balance law for particle-wise conservation of linear momentum.

Balance Law of Angular Momentum

Similar to the principle of linear momentum a balance law for angular momentum can be derived. When the definition of angular momentum from classical mechanics is considered the definition can be generalized to a continuum. This definition says that angular momentum is the moment of momentum about an axis. This, in turn, means that it is the product of the linear momentum and the perpendicular distance from the axis of its line of action. The generalization to a continuum then reads

$$\mathbf{L} = \iiint\limits_{\Omega^{\mathcal{P}}} \mathbf{r}\times(\rho\mathbf{v})\,d\mathbf{x},$$

where $\mathbf{r}$ represents the vector pointing from a fixed, arbitrary point in space to $\mathbf{x}$. In the following, the fixed arbitrary point will be (without loss of generality) chosen as the origin, such that $\mathbf{r} = \mathbf{x}$. Analogously to the mathematical transformation in Equation (2.10), the total time derivative of $\mathbf{L}$ representing the moment of force, *i.e.*, torque, can be expressed by

$$\begin{aligned}
\dot{\mathbf{L}} &= \iiint\limits_{\Omega^{\mathcal{P}}} \mathbf{x}\times(\rho\mathbf{v})d\mathbf{x} \\
\text{Reynold's theorem} \rightarrow &= \iiint\limits_{\Omega^{\mathcal{P}}} \left[\frac{d(\mathbf{x}\times(\rho\mathbf{v}))}{dt} + (\mathbf{x}\times(\rho\mathbf{v}))(\nabla\cdot\mathbf{v})\right] d\mathbf{x} \\
&= \iiint\limits_{\Omega^{\mathcal{P}}} \left[\rho\underbrace{\mathbf{v}\times\mathbf{v}}_{=0} + \mathbf{x}\times(\dot{\rho}\mathbf{v}+\rho\dot{\mathbf{v}}) + (\mathbf{x}\times(\rho\mathbf{v}))(\nabla\cdot\mathbf{v})\right] d\mathbf{x} \\
&= \iiint\limits_{\Omega^{\mathcal{P}}} \left[\mathbf{x}\times(\rho\dot{\mathbf{v}}) + \underbrace{(\dot{\rho}+\rho\nabla\cdot\mathbf{v})}_{\substack{=0 \\ (\textit{cf.}\text{ Equation (2.8)})}}(\mathbf{x}\times\mathbf{v})\right] d\mathbf{x} \\
&= \iiint\limits_{\Omega^{\mathcal{P}}} \mathbf{x}\times(\rho\dot{\mathbf{v}})d\mathbf{x}.
\end{aligned}$$

Building on the concept of forces, it can be assumed that the total torque $\boldsymbol{\tau}$ acting on a portion of a body is

$$\boldsymbol{\tau} = \iiint\limits_{\Omega^{\mathcal{P}}} \mathbf{r}\times\mathbf{b}d\mathbf{x} + \oiint\limits_{\partial\Omega^{\mathcal{P}}} \mathbf{r}\times\mathbf{t}dS.$$

Similar to Newton's second law of motion, *Euler's second law of motion* states that the rate of change of angular momentum about a point equals the total external torque in the system. Applying this law to the continuum formulation yields

$$\dot{\mathbf{L}} = \boldsymbol{\tau}$$
$$\Leftrightarrow$$
$$\iiint\limits_{\Omega^{\mathcal{P}}} \mathbf{x} \times (\rho \dot{\mathbf{v}})\, d\mathbf{x} = \iiint\limits_{\Omega^{\mathcal{P}}} \mathbf{x} \times \mathbf{b} d\mathbf{x} + \oiint\limits_{\partial\Omega^{\mathcal{P}}} \mathbf{x} \times \mathbf{t} dS. \tag{2.13}$$

This formulation then represents the *global balance law of angular momentum*. In order to localize this expression, the surface integral part of Equation (2.13) has to be transformed into a volume integral, *i.e.*,

$$\begin{aligned}\oiint\limits_{\partial\Omega^{\mathcal{P}}} \mathbf{x} \times \mathbf{t} dS &= \oiint\limits_{\partial\Omega^{\mathcal{P}}} \mathbf{x} \times (\boldsymbol{\sigma}^T \mathbf{n}) dS \\ &= \oiint\limits_{\partial\Omega^{\mathcal{P}}} \left[\mathbf{x}^{\times} \boldsymbol{\sigma}^T\right] \mathbf{n} dS \\ &= \iiint\limits_{\Omega^{\mathcal{P}}} \nabla \cdot \left[\mathbf{x}^{\times} \boldsymbol{\sigma}^T\right] d\mathbf{x} \\ &= \iiint\limits_{\Omega^{\mathcal{P}}} \left[\boldsymbol{\sigma}^{\times} + \mathbf{x} \times (\nabla \cdot \boldsymbol{\sigma}^T)\right] d\mathbf{x},\end{aligned}$$

where the operator $(\cdot)^{\times}$ is defined as $[\mathbf{x}^{\times}]_{ik} = \sum_j \varepsilon_{ijk} x_j$ for vectors and $[\boldsymbol{\sigma}^{\times}]_i = \sum_j \sum_k \varepsilon_{ijk} \sigma_{jk}$ for tensors with ε_{ijk} being the *Levi-Civita tensor*. When the volume integral is plugged back into Equation (2.13) the law can be localized and simplifies to

$$\iiint\limits_{\Omega^{\mathcal{P}}} \left[-\boldsymbol{\sigma}^{\times} + \mathbf{x} \times \underbrace{\left(\rho \dot{\mathbf{v}} - \nabla \cdot \boldsymbol{\sigma}^T - \mathbf{b}\right)}_{\substack{=0 \\ \text{(\textit{cf.} Equation (2.12))}}} \right] d\mathbf{x} = 0 \quad \Rightarrow \quad \boldsymbol{\sigma}^{\times} = \mathbf{0}$$

$$\boldsymbol{\sigma}^{\times} = \mathbf{0} \quad \Leftrightarrow \quad \boldsymbol{\sigma} = \boldsymbol{\sigma}^T.$$

Therefore, the evaluation of the local balance law for angular momentum proofs that the Cauchy stress tensor has to be symmetric.

At this point I would like to point out that there are indeed cases, *i.e.*, if there are other sources for external torque or if coupling stress is assumed, where the stress tensor is not symmetric. One class of these models is the micropolar model [Łukaszewicz 1999] that has also been applied to fluid simulation in the context of computer graphics by Bender *et al.* [2017] to simulate turbulent flow.

Finally, the acquired local balance laws are summarized below.

$$\boxed{\begin{array}{rrcl} \text{(Continuity equation)} & \dot{\rho} & = & -\rho \nabla \cdot \mathbf{v} \\ \text{(Linear momentum conservation)} & \rho \dot{\mathbf{v}} & = & \nabla \cdot \boldsymbol{\sigma} + \mathbf{b} \\ \text{(Angular momentum conservation)} & \boldsymbol{\sigma} & = & \boldsymbol{\sigma}^T \end{array}}$$

Balance Laws in Reference Formulation

In some applications it is advantageous to formulate the derived field equations with respect to a reference configuration. This is moreover required when simulations in Lagrangian coordinates are desired. In the following, a body $\mathcal{B}$ with reference configuration $\mathcal{X}_R$ and time-varying current configuration $\mathcal{X}$ with $\Omega_R = \mathcal{X}_R(\mathcal{B})$ and $\Omega = \mathcal{X}(\mathcal{B})$ will be considered. Moreover, $\boldsymbol{\xi} \in \Omega_R$ and $\mathbf{x} \in \Omega$ will denote spatial positions of material particles in reference and current configuration, respectively. In order to reformulate the local balance laws with respect to a reference configuration, transformation rules for line, oriented surface area, and volume elements are required.

A relation for line elements is already known from Equation (2.1). To derive a transformation for oriented surface area elements, they first have to be defined and subsequently a relation to line elements has to be made. A surface element in reference and current configuration will be denoted dS_R and dS, respectively. An oriented surface element is then the product of the surface area element with its surface normal, *i.e.*, $\mathbf{n}_R dS_R$ and $\mathbf{n}dS$. Such an oriented surface element can further be represented by two linearly independent line element vectors $d\mathbf{x}^{(1)} = \mathbf{F}d\boldsymbol{\xi}^{(1)}$ and $d\mathbf{x}^{(2)} = \mathbf{F}d\boldsymbol{\xi}^{(2)}$ in the surface's tangent space. This yields the following relations: $\mathbf{n}dS = d\mathbf{x}^{(1)} \times d\mathbf{x}^{(2)}$ and $\mathbf{n}_R dS_R = d\boldsymbol{\xi}^{(1)} \times d\boldsymbol{\xi}^{(2)}$. A mathematical transformation can then be directly derived, *i.e.*,

$$d\mathbf{x}^{(1)} \times d\mathbf{x}^{(2)} = (\mathbf{F}d\boldsymbol{\xi}^{(1)}) \times (\mathbf{F}d\boldsymbol{\xi}^{(2)}) = J\mathbf{F}^{-T}(d\boldsymbol{\xi}^{(1)} \times d\boldsymbol{\xi}^{(2)})$$
$$\Leftrightarrow$$
$$\mathbf{n}dS = J\mathbf{F}^{-T}\mathbf{n}_R dS_R,$$

with $J = \det \mathbf{F}$. A transformation between volume elements $dv_R = d\boldsymbol{\xi}^{(1)} \cdot (d\boldsymbol{\xi}^{(2)} \times d\boldsymbol{\xi}^{(3)})$ and $dv = d\mathbf{x}^{(1)} \cdot (d\mathbf{x}^{(2)} \times d\mathbf{x}^{(3)})$ can be derived in a similar way, *i.e.*,

$$d\mathbf{x}^{(1)} \cdot (d\mathbf{x}^{(2)} \times d\mathbf{x}^{(3)}) = (\mathbf{F}d\boldsymbol{\xi}^{(1)}) \cdot \left[(\mathbf{F}d\boldsymbol{\xi}^{(2)}) \times (\mathbf{F}d\boldsymbol{\xi}^{(3)})\right] = J\ \left[d\boldsymbol{\xi}^{(2)} \cdot (d\boldsymbol{\xi}^{(2)} \times d\boldsymbol{\xi}^{(3)})\right]$$
$$\Leftrightarrow$$
$$dv = Jdv_R.$$

With these transformations the global balance laws can be reformulated in reference coordinates and subsequently localized. Let, again, $\mathcal{P} \subset \mathcal{B}$ be a subset of $\mathcal{B}$ with $\Omega^{\mathcal{P}} = \mathcal{X}(\mathcal{P})$ and $\Omega_R^{\mathcal{P}} = \mathcal{X}_R(\mathcal{P})$. Then,

$$\iiint\limits_{\Omega^{\mathcal{P}}} \rho\dot{\mathbf{v}}d\mathbf{x} = \iiint\limits_{\Omega_R^{\mathcal{P}}} \underbrace{\rho J}_{\rho_R}\dot{\mathbf{v}}d\boldsymbol{\xi}, \quad \iiint\limits_{\Omega^{\mathcal{P}}} \mathbf{b}d\mathbf{x} = \iiint\limits_{\Omega_R^{\mathcal{P}}} \underbrace{\mathbf{b}J}_{\mathbf{b}_R}d\boldsymbol{\xi}$$

$$\iiint\limits_{\Omega^{\mathcal{P}}} \nabla \cdot \boldsymbol{\sigma}d\mathbf{x} = \oiint\limits_{\partial\Omega^{\mathcal{P}}} \mathbf{t}dS = \oiint\limits_{\partial\Omega^{\mathcal{P}}} \boldsymbol{\sigma}\mathbf{n}dS = \oiint\limits_{\partial\Omega_R^{\mathcal{P}}} \underbrace{J\boldsymbol{\sigma}\mathbf{F}^{-T}}_{\mathbf{P}}\mathbf{n}_R dS_R = \iiint\limits_{\Omega_R^{\mathcal{P}}} \nabla \cdot \mathbf{P}d\boldsymbol{\xi}.$$

Here, ρ_R and $\mathbf{b}_R$ denote the reference density and body forces in reference space. Moreover, $\mathbf{P} = J\boldsymbol{\sigma}\mathbf{F}^{-T}$ is referred to as the *first Piola-Kirchhoff stress tensor*. Interpreting stress tensors as linear operator $\boldsymbol{\sigma} : \mathcal{S} \to \mathbb{R}^3$, *e.g.*, $\mathbf{t} = \boldsymbol{\sigma}\mathbf{n}$, the Cauchy stress tensor relates normals in the current configuration with tractions in the current configuration. To give $\mathbf{P}$ a similar interpretation; $\mathbf{P}$ relates normals in the reference configuration with normals in the current configuration, *i.e.*, $\mathbf{t} = \mathbf{P}\mathbf{n}_R$. Also a stress tensor $\mathbf{S}$ can be found that purely acts in the reference configuration such that $\mathbf{t}_R = \mathbf{S}\mathbf{n}_R$. It can be shown that $\mathbf{S} = J\mathbf{F}^{-1}\boldsymbol{\sigma}\mathbf{F}^{-T} = \mathbf{F}^{-1}\mathbf{P}$, where $\mathbf{S}$ is referred to as *second Piola-Kirchhoff stress tensor*.

As a result the law for point-wise linear momentum conservation can be reformulated in reference space, *i.e.*,

$$\boxed{\rho_R \dot{\mathbf{v}} = \nabla \cdot \mathbf{P} + \mathbf{b}_R} \tag{2.14}$$

All simulation approaches presented in this thesis will be of Lagrangian nature and therefore be based on the reference formulation of the balance of linear momentum presented in Equation (2.14). It holds independently of the nature of the simulated matter including solids, liquids, and gases.

2.4 Constitutive Laws for Solids and Fluids

As previously mentioned, the equation for linear momentum conservation is independent of the simulated material. For that reason, the material properties have to be encoded in the model. The mechanical properties of a certain material are characterized by relations between mechanical stress, in this model expressed by the stress tensors, and external stimuli. For elastically deformable objects the external stimulus is strain or more general deformation while Newtonian fluids generate stress from compression or from the shear strain rate. Model equations that relate the external stimuli with the mechanical stress are referred to as constitutive equations. In the following, several constitutive models for isotropic elastic deformations and the constitutive law for Newtonian fluids will be presented.

Isotropic Hyperelasticity

All constitutive laws for elastic objects presented in this work will be derived from potentials. Such models are referred to as *hyperelastic constitutive models* or *Green elastic materials*. Therefore, it is assumed that a function $\Psi = \Psi(\mathbf{F})$ exists that maps a deformation to an energy density. Given a deformation represented by a deformed configuration $\mathcal{X}$ at time t, the elastic potential (or elastic energy) of a solid object is

$$E = \iiint\limits_{\Omega_R} \Psi(\mathbf{F}(\boldsymbol{\xi}))d\boldsymbol{\xi},$$

such that the first Piola-Kirchhoff stress tensor is

$$\mathbf{P}(\mathbf{F}) = \frac{\partial \Psi}{\partial \mathbf{F}}.$$

St. Venant-Kirchhoff Model

The St. Venant-Kirchhoff model is the simplest material model and is, with the free energy function being quadratic in $\mathbf{E}$ and therefore quartic in the deformation gradient $\mathbf{F}$, a polynomial model. The elastic energy density is then defined as

$$\Psi(\mathbf{F}) = \mu \mathbf{E} : \mathbf{E} + \frac{\lambda}{2} \operatorname{tr}(\mathbf{E}),$$

where the material parameters μ and λ are referred to as *Lamé parameters*. Alternatively, a fourth-order tensor $\mathfrak{C}$ with $\mathfrak{C}_{ijkl} = \lambda\delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})$ can be defined such that $\Psi(\mathbf{F}) = \mathbf{E} : \mathfrak{C}\,\mathbf{E}$. The according first Piola-Kirchhoff stress tensor is then

$$\mathbf{P}(\mathbf{F}) = \mathbf{F}(2\mu\mathbf{E} + \lambda \operatorname{tr}\mathbf{E}\ \mathbb{1}).$$

Because the physical meaning of the Lamé parameters is not obvious, they are often expressed by the engineering parameters, *i.e., Young's modulus* $k = \frac{\mu(3\lambda+2\mu)}{\lambda+\mu}$ and *Poisson's ratio* $\nu = \frac{\lambda}{2(\lambda+\mu)}$. Then, the physical interpretation is that $k > 0$ represents the general stiffness of the material whereas $-1 < \nu < \frac{1}{2}$ stands for the amount of lateral contraction or expansion. Further, a material is not affected by lateral effects for $\nu = 0$ and is incompressible in the limit $\nu \to \frac{1}{2}$. A problem of this model is that it is not polyconvex [RAOULT 1986]. Therefore, the associated energy E has no unique minimizer.

Linear Elasticity or (Isotropic) Generalized Hookean Model

In order to formulate a linear constitutive model for small deformations, the St. Venant-Kirchhoff model can be linearized. Then, the energy density function is solely based on the linearized strain, *i.e.,*

$$\Psi(\mathbf{F}) = \mu\boldsymbol{\epsilon} : \boldsymbol{\epsilon} + \frac{\lambda}{2}\operatorname{tr}(\boldsymbol{\epsilon})$$

with the corresponding stress tensor

$$\mathbf{P}(\mathbf{F}) = 2\mu\boldsymbol{\epsilon} + \lambda\operatorname{tr}(\boldsymbol{\epsilon})\,\mathbb{1}.$$

Again, it should be noted that this energy density function is <u>not</u> invariant under rigid body rotations. This means, that if the corresponding body is subject to rigid body rotations, the energy will be non-zero and a stress will emerge within the body.

Corotated Linear Elasticity

In order to formulate a constitutive model that is as linear as possible but does not yield stresses when the simulated body is subject to rigid body rotations, a corotated linear constitutive model can be used. The term "linear" in the model's name is, however, misleading as the material model is non-linear due to the required polar decomposition of the deformation gradient to determine the right stretch tensor $\mathbf{U}$ and the rotation matrix $\mathbf{R}$. The energy density function is then defined by

$$\Psi(\mathbf{F}) = \mu\mathbf{E}_{\text{Biot}} : \mathbf{E}_{\text{Biot}} + \frac{\lambda}{2}\operatorname{tr}^2(\mathbf{E}_{\text{Biot}})$$

with the corresponding stress tensor

$$\mathbf{P}(\mathbf{F}) = \mathbf{R}\left(2\mu\mathbf{E}_{\text{Biot}} + \lambda\operatorname{tr}(\mathbf{E}_{\text{Biot}})\,\mathbb{1}\right).$$

The model is called "corotated" because the used strain measure, *i.e.,* the Biot strain tensor, can be interpreted as linearized strain that is rotated from current into reference configuration such that no rigid body rotations are evident in the final measure.

Governing Equations for Elastic Solids

Within the last sections all required equations and boundary conditions to simulate an elastic solid in Lagrangian coordinates were derived. These are the balance law of linear momentum, a hyperelastic constitutive law, and kinematic equations as well as traction and displacement boundary conditions, and initial displacement conditions. Then, the resulting *Mixed Initial-Boundary Value Problem* (IBVP) is

$$\boxed{\begin{array}{rrcll}
\text{(Linear momentum conservation)} & \rho_R \ddot{\mathbf{u}} & = & \nabla \cdot \mathbf{P} + \mathbf{b}_R & \\
\text{(Constitutive model)} & \mathbf{P} & = & \frac{\partial \Psi(\mathbf{F})}{\partial \mathbf{F}} & \\
\text{(Kinematic equation)} & \mathbf{F} & = & \mathbb{1} + \frac{\partial \mathfrak{U}}{\partial \boldsymbol{\xi}} & \\
& & & & \\
\text{(Initial displacement condition)} & \mathbf{u} & = & \mathbf{u}_0 \quad \text{on} & \Omega \text{ at } t = 0 \\
\text{(Initial velocity condition)} & \mathbf{v} & = & \mathbf{v}_0 \quad \text{on} & \Omega \text{ at } t = 0 \\
\text{(Dirichlet boundary condition)} & \mathbf{u} & = & \mathbf{u}_\mathrm{D} \quad \text{on} & \partial\Omega_\mathrm{D} \\
\text{(Neumann boundary condition)} & \mathbf{t} & = & \mathbf{t}_\mathrm{N} \quad \text{on} & \partial\Omega_\mathrm{N}
\end{array}}$$

where $\partial\Omega_D$ and $\partial\Omega_N$ with $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$ denote portions of the domain's boundary where essential and natural boundary conditions are implied, respectively. The displacement boundary condition $\mathbf{u}_D$ is part of the problem description (so not part of the solution) and prescribes that the affected boundary is either fixed, *i.e.*, when $\mathbf{u}_D = \mathbf{0}$, or that certain boundary regions follow a given trajectory. Similarly, $\mathbf{t}_N$ is also part of the problem description and represents external boundary tractions exerted on the system.

Incompressible Newtonian Fluids

A Newtonian fluid follows an idealized constitutive model that approximates the relation between a fluid's motion and the occurring mechanical stress. It further makes several assumptions about this relation. Firstly, when the fluid rests, the stress within the fluid is assumed to be a hydrostatic pressure. Secondly, it assumes that there is a linear relation between the Cauchy stress tensor and the linearized strain rate. Thirdly, no shear stresses are assumed to occur when the fluid undergoes rigid body motion. Finally, the fluid is assumed to behave isotropically and, therefore, a material particle responds directionally independent to deformation, strain-rate or other internal/external influences. The corresponding constitutive model is then

$$\boldsymbol{\sigma} = -p\, \mathbb{1} + 2\mu\dot{\boldsymbol{\epsilon}},$$

where p denotes the hydrostatic pressure and $\mu > 0$ is a material parameter usually referred to as *dynamic viscosity*. Plugging the constitutive law into the law of linear momentum conservation (2.12) yields

$$\begin{aligned}
\rho\dot{\mathbf{v}} &= \nabla \cdot \boldsymbol{\sigma} + \mathbf{b} \\
&= \nabla \cdot (-p\mathbb{1} + 2\mu\dot{\boldsymbol{\epsilon}}) + \mathbf{b} \\
&= -\nabla p + 2\mu\nabla \cdot \left(\frac{1}{2}\left(\nabla\mathbf{v} + \nabla\mathbf{v}^T\right)\right) + \mathbf{b} \qquad (2.15)\\
&= -\nabla p + \mu\left(\nabla \cdot \nabla\mathbf{v} + \nabla \cdot \nabla\mathbf{v}^T\right) + \mathbf{b} \\
&= -\nabla p + \mu\left(\Delta\mathbf{v} + \nabla(\nabla \cdot \mathbf{v})\right) + \mathbf{b}.
\end{aligned}$$

In the context of this thesis all fluids are assumed to be incompressible. This is an idealization but holds for most fluids that humans interact with in daily life presuming moderate speed of motion and moderate external influences. Incompressibility is enforced if, and only if, the density at every material particle is constant. The according mathematical constraint $\rho = \mathrm{const}$ will be referred to as *constant density condition*. This also implies that the density's rate of change equals zero, *i.e.,* $\dot{\rho} = 0$. Considering the identity given by Equation (2.9), this in turn means that the velocity field must be divergence-free because $\rho > 0$ on $\mathcal{B}$. Mathematically, this implies the velocity level constraint $\nabla \cdot \mathbf{v} = 0$ that will be referred to as *divergence-free condition* in the following. Plugging the divergence-free condition into Equation (2.15) yields the *incompressible Navier-Stokes equations*

$$\boxed{\begin{aligned} \rho\dot{\mathbf{v}} &= -\nabla p + \mu\Delta\mathbf{v} + \mathbf{b} \\ \rho = \mathrm{const} \quad &\Leftrightarrow \quad \nabla \cdot \mathbf{v} = 0. \end{aligned}} \tag{2.16}$$

Please note that no relation between kinematic quantities and the pressure p is given here. While some models rely on a state equation the pressure for an incompressible fluid is implicitly defined as the constant density condition has to be satisfied. Mathematically, p can also be interpreted as Lagrange multiplier that ensures that incompressibility is satisfied in the equations of motion. The quantities pressure and velocity can also be linked by taking the divergence of the first equation and by plugging the result into the divergence-free condition. This eventually results in the so-called *Pressure Poisson Equation* (PPE) that has the form $\Delta p = f(\mathbf{v})$.

By prescribing an initial velocity and (if existent) an initial free surface location and non-penetration boundary conditions again yields an IBVP

$$\boxed{\begin{array}{rrcll} \text{(Linear momentum conservation)} & \rho\dot{\mathbf{v}} & = & -\nabla p + \Delta\mathbf{v} + \mathbf{b} & \\ \text{(Incompressibility)} & \nabla \cdot \mathbf{v} & = & 0 & \\ & & & & \\ \text{(Initial position condition)} & \mathbf{x} & = & \mathbf{u}_0 \quad \text{on} & \Omega \text{ at } t = 0 \\ \text{(Initial velocity condition)} & \mathbf{v} & = & \mathbf{v}_0 \quad \text{on} & \Omega \text{ at } t = 0 \\ \text{(Non-penetration condition)} & \mathbf{v} \cdot \mathbf{n} & = & 0 \quad \text{on} & \partial\Omega_{\text{Wall}}, \end{array}}$$

where $\partial\Omega_{\text{Wall}}$ denotes the portion of the fluid domain boundary that is in contact with a static solid. The model further states that the fluid can move freely on free surfaces that are not in contact.

2.5 Summary

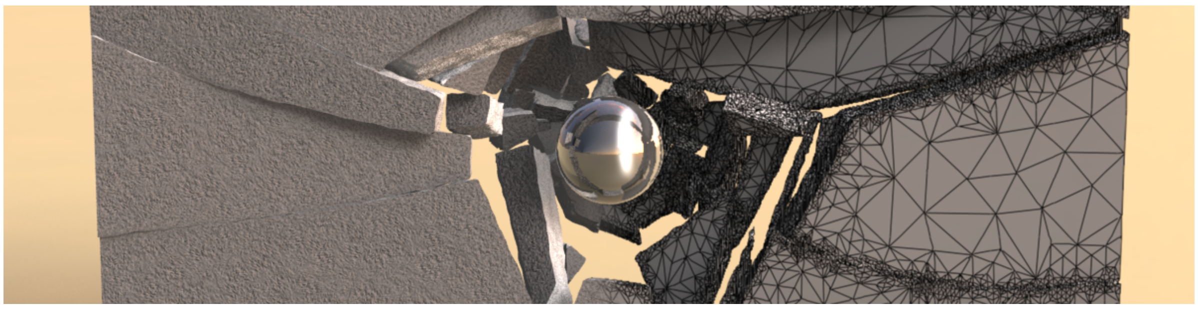
In this chapter, continuum mechanical models that describe idealized solids and fluids were presented. The basis build the laws of mass and linear momentum conservation that can be combined to get a PDE describing the dynamic motion of the considered matter. The derivation of the conservation law of angular momentum showed that the Cauchy stress tensor has to be symmetric given the stated assumptions. In order to describe the equations of motion with respect to the reference configuration, the local linear momentum balance law was transformed respectively. Finally, several hyperelastic constitutive laws for deformable solids and the Newtonian constitutive law for fluids were derived in order to formulate IBVPs describing specific elasticity or fluid dynamics problems. However, in by far most cases the IBVPs are not analytically solvable and therefore require the computation of a numerical solution. In order to acquire a numerical solution, the differential operators in space and time have to be discretized. In the following chapters numerical simulation methods based on continuum mechanical models to simulate complex physical phenomena will be presented.

Part *I*

Cutting and Fracture of Deformable Solids

Chapter 3

Adaptive Tetrahedral Meshes for Brittle Fracture Simulation



The physically-based simulation of deformable solids has been an active research topic for three decades in the field of computer graphics. Because countless works focus on either modeling deformation based on continuous, discrete, or even purely geometric spatial models, the general topic of simulating elastic deformation in a physically plausible manner can be perceived as solved problem. Recent research on deformable bodies can be generally divided into two directions. The first direction focuses on improving robustness, stability, and computational performance using novel space and time discretization approaches or modern numerical solvers. The second direction is concerned with mathematical modeling of complex physical phenomena such as viscosity, plasticity, thermodynamic effects, *etc*. Further examples for physical phenomena that have been modeled in the computer graphics community are cutting and fracture of deformable solids. These examples are moreover very prominent as according mathematical models are required for the visual animation of destruction and complex close-up scenes in special effects for Hollywood movies, purely animated films, or computer games. Also in the application area of training simulators, there is a great demand for methods to simulate cutting.

A simple criterion for failure in brittle materials can be based on the mechanical stress within the object. Therefore, it is required to perform a numerical analysis in order to determine when and where a crack occurs and how it propagates. Moreover, the spatial discretization of the elasticity IBVP has to be adapted due to the crack propagation such that the arising discontinuity is sufficiently represented. For that reason, the discretization should be very fine in regions experiencing high stresses and along the crack path. However, the regions that will experience high stresses and the location of the cracks are not known a-priori.

In this chapter, a method that employs a novel, reversible spatially adaptive discretization method based on tetrahedral meshes for the simulation of brittle fracture is presented. The approach requires an initial (preferably coarse) input mesh and treats solid objects as rigid bodies as long as they are not subjected to collisions with other objects. Subsequent to a collision a stress analysis is performed and the underlying mesh is successively adapted guided by a stress-based criterion. In this way, highly-stressed regions, *i.e.,* stress concentrations, are accurately determined and the potential crack origins can be located. The adaption scheme is designed to preserve the quality of the input mesh to a large extent. Moreover, it has the advantages that it does not alter the boundary geometry, such that geometric features are maintained, and that it is exclusively based on topological operations. The algorithm is able to generate multiple cracks from a single collision while an approach to suppress excessive shattering in highly stressed regions is incorporated. In order to increase the performance of future stress analyses the previous refinement is partially reversed. The realism of the results produced by the method is demonstrated using several complex scenarios involving hundreds of collisions that produce thousands of fragments.

3.1 Introduction

In our natural environment solid objects fracture when subject to collisions or other sources of external impulses or forces. In the human perception this is a natural phenomenon and part of our everyday experience. However, in the context of physical simulation this phenomenon is very complex and hard to model mathematically. On the way to model our world in virtual environments, it is necessary to develop physically plausible models to improve visual realism. Therefore the simulation of fracture is an important research topic in the field of computer graphics.

The fracture behavior of materials can be roughly categorized into two classes: brittle and ductile fracture. While the case where an object experiences moderate to heavy plastic deformations before the material fails and tears is referred to as ductile fracture, material failure that occurs subsequent to a negligible amount of purely elastic deformation is referred to as brittle fracture. Typical examples for materials undergoing brittle fracture are pottery, concrete, stone, and glass.

In the field of computer graphics, many approaches to simulate brittle fracture rely on a numerical analysis to determine the stress field within a solid. Most of these methods are based on the *Finite Element Method* (FEM) which has proven to be an effective tool for this purpose, *e.g.,* [O'Brien and Hodgins 1999; M. Müller *et al.* 2001; Bao *et al.* 2007]. All of the mentioned methods spatially discretize the IBVP for elasticity using linear Lagrange polynomials on a static tetrahedral mesh and initiate a crack when the tensile stress associated with a tetrahedron exceeds a certain threshold. As a consequence, the spatial resolution of the mesh heavily influences the number and locations of potential crack origins. Also the accuracy of the obtained stress field is dependent on the resolution of the tetrahedral mesh. Therefore, coarse meshes do not provide the desired accuracy to accurately locate stress concentrations while an analysis on high-resolution meshes is computationally expensive. As we are only interested in regions where high tensile stresses occur, it is beneficial to locally refine the mesh in these regions. This observation motivates the proposition to develop a novel spatially adaptive method. However, spatial adaption of tetrahedral discretizations is in general non-trivial. As continuous Galerkin FEMs require that the spatial approximation to the IBVP is C^0 continuous everywhere, the refinement strategy has to preserve the conformity of the mesh. This means that the mesh must not contain T-junctions, also called "hanging nodes", in order to avoid the requirement for further treatments in the discretization. The mesh is also supposed to stay conforming with geometric features of the boundary geometry such as sharp edges or corners. Moreover, it is beneficial if the refinement is (uniquely) reversible in order to coarsen the mesh in regions where the high-resolution is not required anymore to save computational resources in the ongoing simulation.

In the following, a novel reversible refinement algorithm for tetrahedral meshes is proposed. The strategy fulfills all stated requirements that are important for the simulation of brittle fracture and the inherent balancing ensures that the quality of the involved tetrahedra decreases only slightly with each refinement level. The algorithm is then used to formulate an adaptive FEM for the accurate determination of stress concentrations within an elastic solid. Using the proposed fracture algorithm cracks are inserted at the locations where the maximum tensile stress occurs and the stress field is subsequently recomputed. As previously discussed, the number of crack origins is highly dependent on the mesh resolution. In the case of highly detailed meshes, this results in excessive shattering of the stressed region. In order to remove this undesired artifact, a region-wise non-maximum suppression is presented. It is moreover possible, that an element repeatedly indicates material failure after a crack already occurred. Therefore, a stress-rate based criterion is proposed to avoid this temporal shattering artifact. To reduce the computational cost of the simulation, stress analyses are only performed when the object collides with other dynamic solids or static boundaries.

For the general dynamic movement the solid is idealized as rigid body in the limit of infinite stiffness. This strategy is not new and was already employed in several works, *e.g.,* M. MÜLLER *et al.* [2001], BAO *et al.* [2007], and GLONDU *et al.* [2013]. As the elastic solid experiences only small deformation before the material fails, the individual tetrahedra do not become degenerate. Therefore, it turned out to be sufficient to perform all adaption operations and to balance the quality of the tetrahedra in the undeformed reference configuration (*cf.* Chapter 2). Additionally, the proposed refinement is computationally very efficient due to its conceptual simplicity and is exclusively based on topological modifications, *i.e.,* split, merge, and flip operations. In order to be able to uniquely reverse the refinement only a small amount of additional data per simplex has to be stored. An interesting property of the refinement scheme is that the reversibility is also independent of the order of operations.

To summarize, the main contributions of the presented approach are the proposition of:

- an adaptive approach for the simulation of brittle fracture for highly detailed determination of crack origins,
- a novel algorithm for reversible spatial adaption of tetrahedral meshes exclusively based on topological operations that preserves geometric features on the mesh boundary and that maintains the quality of the mesh to a good extent, and
- a simple yet effective approach to prevent spatial and temporal shattering artifacts based on a non-maximum suppression and a stress-rate dependent fracture criterion.

3.2 Related Work

In this section, publications closely related to the presented approach will be discussed. As this work contributes to the field of brittle fracture simulation and mesh adaption, the discussion is organized respectively.

Fracture Simulation

Methods for the physically-based simulation of fracture have been developed and investigated for nearly three decades, although not many articles were published towards the end of the last century. In the pioneering work of TERZOPOULOS and FLEISCHER [1988] inelastic deformable objects are discretized using finite difference schemes or finite elements. They employ a constitutive model based on controlled-continuity generalized spline kernels which are weighted by weighting functions. They further use a stress-based fracture criterion such that the weighting functions are nullified when the internal stress exceeds a certain threshold. The nullification then causes the elastic potential along the crack path to vanish which physically means that the material is separated. For discrete elements, this means that they simply vanish. A few years later, NORTON *et al.* [1991] proposed a method that simulates brittle fracture on the basis of a discrete mass-spring system. They prescribe a spatially varying 'breakage threshold' field and remove springs when the force between two connected mass points exceeds the threshold. The spatial variation is intended to account for material defects such that cracks are initiated in defect regions more frequently. It took several years until the research on the physically based simulation of fracture was revived by O'BRIEN and HODGINS [1999] and experienced actual innovations. They simulate an elastically deformable object based on a linear elastic continuum model and discretize the resulting IBVP using a linear FEM on a tetrahedral mesh and using an explicit time integration scheme. In each time step they compose a so-called separation tensor from the discrete forces acting on each element and subsequently determine the amount and direction of the greatest tensile stress by means of the tensor's largest eigenvalue and corresponding eigenvector. If the tensile stress exceeds a certain threshold, the corresponding discrete element is remeshed such that the plane orthogonal to the eigenvector is explicitly captured within the tetrahedron. However, disadvantageous is that the local remeshing approach quickly leads to deteriorated elements and that the number of additional vertices increases drastically. The approach still places a milestone in the physically based simulation of fracture as it was the first to explicitly account for the direction of newly originating cuts and to guarantee mass conservation by remeshing of the affected area. The method was later extended to model ductile fracture by O'BRIEN *et al.* [2002] and a simplified version was also adopted by PARKER and O'BRIEN [2009] for simulating fracture in real-time. SMITH *et al.* [2001] represent a brittle object using a tetrahedral mesh where each individual tetrahedron represents a single rigid body and constrain face-adjacent tetrahedra rigidly. The object is then simulated as rigid body and the constraint system is then solved using a quasistatic simulation subsequent to a collision. In this equation system the constraint forces are determined via Lagrange multipliers. If the constraint force exceeds a certain threshold the constraint is simply removed. Due to the simplicity of the approach the method is much faster compared to the approach of O'BRIEN and HODGINS [1999] but, because of the lack of the model's physical justification and because of the absence of mesh adaption, not comparable in terms of the quality of results. A method combining the advantages of the last two discussed methods was proposed by M. MÜLLER *et al.* [2001]. They employ a quasistatic, and therefore time-independent, model, discretize the resulting *Boundary Value Problem* (BVP) using a linear FEM, and compute an impact radius for fracturing multiple tetrahedra at once in order to provide a certain independence between mesh resolution and fracture behavior. However, they do not remesh the affected regions but only split the tetrahedra on existing interior faces. Additionally, they have to anchor the object in the location of impact to guarantee that the BVP has a unique solution. In order to improve the visual appearance of the cracks while avoiding complex remeshing Molino *et al.* [2004] present a *Virtual Node Algorithm* (VNA) that duplicates cracked elements and adds virtual nodes for the duplicates. Exploiting this technique Bao *et al.* [2007] presented a VNA-based, quasistatic method to simulate fracture of thin shells and volumetric objects. They also remove the necessity to anchor the objects by eliminating the null space in the arising equation system. Another approach based on the VNA was proposed by Glondu *et al.* [2013]. Their goal was to achieve a real-time fracture simulation while keeping the IBVP time-dependent. Therefore, they perform a modal analysis on a linear *Finite Element* (FE) discretization such that they can compute an analytic solution for the spatially discrete system. Since they also employ a rigid body simulation for the general dynamic movement, they estimate the contact durations of collisions and provide a contact force model for this duration. A disadvantage of this approach is that the modal analysis is invalidated when the solid fractures or when the discretization is altered in any way, *e.g.*, due to mesh adaptions. Computing further modal analyses for the resulting fragments is, however, unfeasible if a simulation at interactive rates is desired. An approach for the simulation of fracture that employs proper remeshing was proposed by Wicke *et al.* [2010]. Their algorithm is able to dynamically improve the quality of a tetrahedral mesh by local application of both topological operations and vertex smoothing. While they build the fracture algorithm entirely on the work of O'Brien and Hodgins [1999], they use their sophisticated remeshing approach to drastically improve the mesh quality after cracks are inserted. Unfortunately, this dynamic local remeshing method is very hard to implement and cannot guarantee to converge quickly to a high quality mesh due to the discrete nature of the problem. They also have to ensure that transferring physical field quantities from the original to the improved mesh does not suffer from excessive numerical diffusion. Another remeshing focused method for physically based simulation of two-dimensional continua was proposed by Busaryev *et al.* [2013]. Instead of locally modifying the mesh due to a crack, they globally remesh the whole domain using a constrained Delaunay approach. While a high quality of the resulting mesh is guaranteed, the approach is computationally intensive as the whole domain must be considered in every adaption. Additionally, the transfer of physical quantities without excessive numerical diffusion poses a problem. Hegemann *et al.* [2013] refrain from explicitly embedding the crack surface into their tetrahedral discretization and rather track regions of undamaged material using discrete level-sets. As their goal is to model ductile fracture they evolve the level-sets based on the shape derivative of the Griffith energy. In contrast to most of the approaches presented towards the year 2014, Levine *et al.* [2014] revisit the usage of mass-spring systems for fracture computation using a peridynamic model. They sample three-dimensional objects with point masses and connect all particles that are neighboring each other in a certain radius. To model dynamic fracture, springs are removed when their strain exceeds a certain threshold.

In contrast to the discussed works using mesh-based discretizations, also meshless approaches were investigated. Pauly *et al.* [2005] discretize the elasticity IBVP using a meshless, moving-least squares based particle discretization. The discontinuity in the PDE's solution implied by a crack is then enforced using a transparency criterion between interacting particles. The approach can moreover be considered to be spatially adaptive as they employ particle resampling to avoid sparsely sampled regions, although they do not use the spatial adaptivity to improve the solution of the stress analysis. Another (partially) meshless method for the simulation of fracture was proposed by Chen *et al.* [2013]. They first discretize the object using a tetrahedral mesh and place a particle sample on each tetrahedron centroid to spatially discretize the elasticity IBVP using SPH . The discontinuities implied by the cracks are then captured by removing connections between particles neighboring over a tetrahedron face.

Besides choosing physical approaches for the simulation of fracture, methods relying on purely geometric decompositions have become popular in the past. By computing a decomposition of the material prior to the simulation, the material is separated on the precomputed interfaces when the initiation of a crack is triggered. Prescoring is generally a very efficient method to animate fracture and therefore desirable for real-time applications but, by far, less realistic since physical factors such as impact point location and impact magnitude subsequent to a collision are neglected. Methods that fall under this category were, *e.g.,* presented by SU *et al.* [2009] and GLONDU *et al.* [2014]. A variant of these geometric approaches was proposed by MÜLLER *et al.* [2013]. However, they use precomputed fracture patterns and decompose the corresponding object respectively. In this way they can alleviate the physical inaccuracy by aligning the pattern with the impact location.

In contrast to all of the discussed approaches, the method presented in this chapter is the first to perform a spatially adaptive stress analysis on tetrahedral meshes for the physically based simulation of brittle fracture. It moreover employs a method to decouple the mesh resolution and the number of originating cracks which consequently avoids excessive spatial and temporal shattering.

Adaptive Tetrahedral Meshes in Physically Based Simulation

In this section spatially adaptive techniques based on tetrahedral meshes for physically based animation are reviewed. A general overview over adaptive simulations in this field can be found in the review of MANTEAUX *et al.* [2017].

Strategies for the adaption of tetrahedral meshes can be generally divided into local and global techniques. Global adaption techniques regenerate a mesh from scratch due to changed requirements. In local adaption techniques a series of operations, *e.g.,* topological splits, merges, or flips or continuous vertex smoothing, is performed on a certain subset of the tetrahedra composing the mesh. While global mesh regeneration approaches are able to guarantee a certain mesh quality dependent on the meshing algorithm, they imply a considerable computational overhead when a higher mesh resolution in local regions is desired. As local refinement is favored for the accurate determination of stress concentrations in fracture simulation, this section focuses on techniques for local adaption.

A local adaption technique based on red-green refinement was proposed by MOLINO *et al.* [2003]. In this approach a level-set implicitly representing the surface of a given input shape and a regular hexahedral grid are employed. Subsequently, each grid cell is decomposed into tetrahedra which are marked as 'red'. A tetrahedron is then refined using a sequence of edge split operatios (see *e.g.,* BANK *et al.* [1983]) resulting in a regular pattern of eight smaller 'red' tetrahedra. In order to keep the resulting mesh conforming, neighboring tetrahedra have to be irregularly subdivided to match the refinement pattern. These irregularly refined tetrahedra are then called 'green'. Although they do not use the refinement technique within a simulation, they locally refine an initial mesh to improve conformance to the given level set while keeping the mesh quality high. A similar approach that uses a grid structure and a level-set as input was proposed by LABELLE and SHEWCHUK [2007]. They use predefined stencils to decompose grid cells into tetrahedra. Moreover, they directly account for partially filled cells and can even guarantee bounds on the dihedral angles of the resulting polyhedra. They also propose a method to construct a graded mesh such that tetrahedra close to the boundary are finer than interior tetrahedra. These level-set based techniques were later applied to the physical simulation of fluids, *e.g.*, by WOJTAN and TURK [2008] or ANDO *et al.* [2013]. A method for the simulation of deformable solids based on a hierarchy of non-nested tetrahedral meshes was proposed by DEBUNNE *et al.* [2001]. They use a linear FE discretization in combination with explicit time integration. In a first step, they assign nodes of finer hierarchies to nodes of their parent hierarchy level. In this way field quantities can be interpolated from finer to coarser levels. Guided by an error based refinement criterion, they can then de-/activate mutually assigned nodes if required. Typically, stiffness parameters of brittle materials are very large resulting in a stiff system of *Ordinary Differential Equation*s (ODEs) which makes implicit time integration inevitable. However, the construction of equation systems for implicit time integration is more than non-trivial in this case as the set of active nodes belonging to different hierarchy levels makes the discretization discontinuous and therefore the evaluation of integrals over the nodal support domains complex. This issue is also ignored in this approach, which makes the (at least spatial) convergence of the method questionable. X. WU *et al.* [2001] proposed a method for the dynamic progressive refinement of tetrahedral meshes for the simulation of deformable solids. Prior to the simulation they perform a series of edge collapse operations to reduce an initial high-resolution mesh and track all of the changes. During the simulation runtime they apply the reverse operations, *i.e.*, vertex splits, in regions indicated by an error estimator. It is, however, unclear how to maintain the quality of the mesh during coarsening and refinement. This is also stated as a limitation by the authors. A less traditional approach for the locally adaptive simulation of deformable bodies was proposed by GRINSPUN *et al.* [2002]. Instead of geometrically refining the tetrahedra within a given mesh, they employ a hierarchical basis to locally add new degrees of freedom improving the accuracy of the simulation. CAPELL *et al.* [2002] followed a similar strategy using hierarchical basis refinement. However, the object has to be volumetrically parametrized which proves to be a non-trivial task. Therefore, they embed the object into a ‘ficticious’ domain that does not conform to the object boundary but is nevertheless easier to parametrize. As they do not account for partially filled tetrahedra, the accuracy of the method is limited. Instead of following static rules for local adaption, KLINGNER and SHEWCHUK [2008] proposed a method to dynamically improve the quality of tetrahedral meshes based on local topological splits, flips, and vertex smoothing. This type of incremental quality improvement was then adopted for the simulation of finite plastic deformations and fracture by WICKE *et al.* [2010]. As already mentioned in the last section, the dynamic local remeshing method is very hard to implement and cannot guarantee to converge quickly to a high quality mesh due to the discrete nature of the quality optimization problem. Among the different existing approaches, the method of BURKHART *et al.* [2010] is most similar to the one presented in this chapter. Inspired by the popular $\sqrt{3}$-subdivision for triangle meshes proposed by KOBBELT [2000], they generalize the approach to volumetric simplicial meshes for the adaptive simulation of elasticity problems in engineering. While the approach yields very good results in terms of the evolution of the surface triangles, the quality of the tetrahedra deteriorates very quickly.

In contrast to most of the discussed approaches, this chapter presents a static subdivision rule for tetrahedral meshes that is solely based on simple topological operations. Tracking the refinement process is therefore trivial and the refinement can be uniquely reversed while the reversion is order independent. Since the refinement is purely based on mesh refinement, it can be directly incorporated into existing finite element solvers (or similar polyhedral based discretization approaches) without any further considerations. The most similar approaches, also employing static local refinement rules, are red-green refinement [MOLINO *et al.* 2003] and the approach of BURKHART *et al.* [2010]. Compared to red-green refinement the number of tetrahedra grows slower in the presented approach and requires no special treatment of neighboring tetrahedra to ensure conformity. Compared to the method of BURKHART *et al.* [2010], the presented method is easily reversible and maintains the quality of the initial mesh to a larger extent.

3.3 Fracture Generation Algorithm

In this section a novel algorithm for the physically based simulation of brittle fracture is presented. The algorithm can be subdivided into several stages. Therefore, the the following description is organized accordingly.

As discussed earlier the phenomenon of brittle fracture arises as a result of large tensile stresses evident in a deformable solid. Because the definition of brittle fracture implies that the corresponding materials deform very slightly before a crack originates, the general dynamic behavior is simulated using a rigid body simulator as long as the object is moving freely and not subject to any collisions. This strategy is advantageous as the movement of the body can be described by only three translational and three rotational degrees of freedom. As this work is rather focused on the numerical stress analysis and crack generation, I would like to refer the reader to the state-of-the-art report of BENDER *et al.* [2014a] for a general overview of the simulation of rigid bodies in the context of computer graphics. As a consequence of a collision with another dynamic or static body, high internal stresses propagating through the material are expected. Hence, a stress analysis is performed in the event of a collision. The strategy to idealize a brittle object as rigid body and to analyze mechanical stresses only subsequent to collisions is not new and has been employed in several previous approaches, *e.g.,* by M. MÜLLER *et al.* [2001], BAO *et al.* [2007], and GLONDU *et al.* [2013].

In the following, a brief overview over the whole fracture generation algorithm is provided. Afterwards a detailed description of every single step is given. In order to realize the simulation of rigid bodies with robust contact handling the open source library Bullet COUMANS [2015] was used. After a collision has been detected a contact duration is estimated. Subsequently, deformations and the resulting internal stresses are determined numerically using a linear finite element analysis. In the course of the analysis, the stress field is inspected for large tensile components. In an iterative procedure regions experiencing large tensile stresses are refined according to the adaption scheme presented in Section 3.4. The stress analysis is then repeated until no further refinement is necessary. To avoid excessive shattering within the object, local maxima in terms of the largest principle stress component are located while tetrahedral elements that experience a strongly negative stress-rate are excluded such that only a single crack originates within a stressed region at once. The fracture criterion used in this algorithm is based on the Rankine hypothesis [D. GROSS and SEELIG 2011]. Thus, it is assumed that a crack initiates if the tensile stress exceeds a certain threshold. If the criterion has been met anywhere within the object, the mesh is separated guided by an implicitly represented fracture surface. The implicit representation is realized using an *Signed Distance Field* (SDF). After a crack propagated through the material, the stressed regions typically relax quickly. Therefore, the tetrahedral mesh is coarsened in order to save valuable computation time in the further process. Finally, the mesh is inspected for topologically disjoint parts. If disjoint parts are evident, each part will be converted into a distinct, topologically connected mesh.

Contact Duration Estimation

In the course of the simulation a rigid body interacts with its surroundings and will be subjected to several collisions. The typical way to resolve these collisions is to apply sudden impulses to the collision pair. As it is the goal to analyze the mechanical stress field over time during a collision, a contact duration has to be estimated. The Hertz model for contact duration estimation of colliding spheres was proposed by JOHNSON [1985]. In order to get a rough estimation of the contact duration the model can be simply generalized to arbitrary objects by replacing the sphere radius with the distance between the contact point $\mathbf{p}_\mathrm{c}$ and the rigid body's center of mass $\mathbf{p}_\mathrm{com}$ resulting in

$$\Delta t_c = c \left(\frac{m^2}{k^2 \|\mathbf{p}_\mathrm{com} - \mathbf{p}_\mathrm{c}\|} \cdot \frac{1}{\mathbf{v}_\mathrm{rel}} \right)^{\frac{1}{5}},$$

where c denotes a user-defined scaling factor, m the rigid body's mass, k Young's modulus and v_rel the relative velocity between the contact points in the direction of the contact normal. For all results generated with this algorithm the scaling constant c was set to one. It should be a valid assumption that the generalization holds for convex objects that have a shape similar to a sphere. However, our results show that the generalization also works well with the fracture generation algorithm for complex non-convex shapes, although the estimation might not be physically valid.

Finite Element Analysis

After the estimation of a contact duration Δt_c, a dynamic finite element analysis using the continuum model presented in Chapter 2 on the corresponding objects is performed to find stress concentrations. Based on the fact that brittle materials undergo very small deformations before they fracture, the usage of a linear strain measure together with the isotropic Hookean constitutive model is justified. This results in the following IBVP

$$\begin{array}{rrcll}
\text{(Linear momentum conservation)} & \rho_R \ddot{\mathbf{u}} & = & \nabla \cdot \mathbf{P} + \mathbf{b}_R & \\
\text{(Constitutive model)} & \mathbf{P} & = & \mathfrak{C}\boldsymbol{\epsilon} & \\
\text{(Kinematic equation)} & \boldsymbol{\epsilon} & = & \frac{1}{2}\left(\mathbf{H} + \mathbf{H}^T\right) & \\
\text{(Initial displacement condition)} & \mathbf{u} & = & \mathbf{0} & \text{on} \quad \Omega \text{ at } t = t_0 \\
\text{(Initial velocity condition)} & \mathbf{v} & = & \mathbf{v}_0 + \boldsymbol{\omega}_0 \times (\mathbf{p}_\mathrm{c} - \mathbf{p}_\mathrm{com}) & \text{on} \quad \Omega \text{ at } t = t_0 \\
\text{(Neumann boundary condition)} & \mathbf{t} & = & \mathbf{t}_\mathrm{col} & \text{on} \quad \Gamma_\mathrm{c},
\end{array} \tag{3.1}$$

where $\mathbf{t}_\mathrm{col}$ denotes the collision traction acting on the region in contact Γ_c. Here, four assumptions are made. Firstly, the colliding solid is assumed to be undeformed prior to the collision resulting in the given initial displacement condition. Secondly, it is assumed that the six degree-of-freedom velocity initialize the point-wise velocity as given in the initial velocity condition. The third assumption is that the object experiences a boundary traction due to the collision force at the impact location. Finally, it is presumed that the object is freely moving and therefore no displacement boundary conditions are prescribed.

In the FEM literature the IBVP is often referred to as the strong form of the problem. In order to discretize the problem using a linear FE discretization the so-called weak form has to be derived prior to a polynomial discretization. The weak form is further defined as an inner product of the PDE with a (non-zero) test function. When a solution to $\mathbf{u}$ is found that fulfills the weak form, it is called a weak solution. This solution does not necessarily have to fulfill the PDE point-wise but only in this weak sense. Therefore, one has to choose a weak solution space and a weak test space that fulfills the minimal smoothness requirements. Let $H^1(\Omega_R)$ denote the Sobolev space of square-integrable and once differentiable functions. Then, one can define a vector-valued weak solution space $\mathbf{U}$ for the displacement and a vector-valued weak test space $\mathbf{W}$ such that

$$\begin{aligned}\mathbf{u} \in \mathbf{U} &:= \left\{ \mathbf{u} \in \left(H^1(\Omega_R)\right)^3 \;\middle|\; \mathbf{u}|_{\partial\Omega_D} = \mathbf{u}_D \right\}\\ \mathbf{w} \in \mathbf{W} &:= \left\{ \mathbf{w} \in \left(H^1(\Omega_R)\right)^3 \;\middle|\; \mathbf{w}|_{\partial\Omega_D} = \mathbf{0} \right\},\end{aligned}$$

where $\mathbf{u}_D$ denotes a predefined displacement on the constrained portion of the boundary $\partial\Omega_D$. This definition includes the requirements that the weak solution space strongly fulfills the essential boundary conditions and that the test space vanishes at the Dirichlet boundary. Taking the L^2-inner product of the PDE (*cf.* 3.1) with the test function on the considered domain yields

$$\iiint\limits_{\Omega_R} \mathbf{w} \cdot \left(\rho_R \ddot{\mathbf{u}} - \nabla \cdot \mathbf{P} - \mathbf{b}_R\right) d\boldsymbol{\xi} = 0.$$

By integrating by parts and by applying the divergence theorem this equation can be transformed into

$$\iiint\limits_{\Omega_R} \mathbf{w} \cdot \left(\rho_R \ddot{\mathbf{u}}\right) d\boldsymbol{\xi} + \iiint\limits_{\Omega_R} \nabla \mathbf{w} : \mathbf{P} d\boldsymbol{\xi} = \oiint\limits_{\partial\Omega_R} \mathbf{w} \cdot \underbrace{\left(\mathbf{P}^T \mathbf{n}\right)}_{\mathbf{t}_R} ds + \iiint\limits_{\Omega_R} \mathbf{w} \cdot \mathbf{b}_R d\boldsymbol{\xi}$$

Because a linear strain measure is employed, the first Piola-Kirchhoff stress tensor $\mathbf{P}$ and the Cauchy stress tensor $\boldsymbol{\sigma}$ do not differ in infinitesimal motions to leading order, *i.e.*, $\mathbf{P} = \boldsymbol{\sigma}$. This also implies that $\mathbf{P} = \mathbf{P}^T$ in this particular case. By exploiting this symmetry property, by plugging in the linear constitutive equation, and by accounting for the fact that the reference boundary traction is only non-zero where natural boundary conditions are defined, the equation transforms into

$$\text{(Weak form)} \quad \iiint\limits_{\Omega_R} \mathbf{w} \cdot \left(\rho_R \ddot{\mathbf{u}}\right) d\boldsymbol{\xi} + \iiint\limits_{\Omega_R} \boldsymbol{\epsilon}(\mathbf{w}) : \mathfrak{C}\boldsymbol{\epsilon}(\mathbf{u}) d\boldsymbol{\xi} = \iint\limits_{\Gamma_c} \mathbf{w} \cdot \mathbf{t}_c ds + \iiint\limits_{\Omega_R} \mathbf{w} \cdot \mathbf{b}_R d\boldsymbol{\xi},$$

where $\boldsymbol{\epsilon}(\mathbf{u}) = \frac{1}{2}\left(\nabla\mathbf{u} + \nabla\mathbf{u}^T\right)$. This is equation is then referred to as the weak form of the problem.

In the event of a collision the rigid body simulator yields a collision impulse $\mathbf{I}_c$ at the contact point $\mathbf{p}_{\text{col}}$ at the current time t_0. Given the definition of an impulse as the integral over a force function $\mathbf{f}_c$ on the time domain, *i.e.*,

$$\mathbf{I}_c = \int\limits_{t_0}^{t_0+\Delta t} \mathbf{f}_c dt, \tag{3.2}$$

one can simply find the force function $\mathbf{f}_c$ such that the equation is fulfilled. In this regard GLONDU *et al.* [2013] modeled the contact force using a smooth sinusodial function that fulfills Equation (3.2). However, it was found that the simulation also yields good results when a constant collision force is used, *i.e.*, $\mathbf{f}_c = \mathbf{I}_c/\Delta t_c$. Describing the collision force as a boundary traction involves the projection of the resultant force onto the affected boundary region. In order to simplify the process, one can replace the Neumann boundary condition with $\mathbf{t}_c = \mathbf{0}$. Then, the contact force $\mathbf{f}_c$ can be applied as body force. If the collision point $\mathbf{p}_c$ coincides with a vertex of the tetrahedral mesh, the respective entries in the body force vector $\mathbf{b}_R$ are set. Otherwise, the collision point lies on a boundary triangle. Then, the force is distributed between the triangle's incident vertices and weighted using the barycentric coordinates of the contact point implied by the triangle. The weak form then reduces to

$$\iiint\limits_{\Omega_R} \mathbf{w} \cdot (\rho_R \ddot{\mathbf{u}})\, d\boldsymbol{\xi} + \iiint\limits_{\Omega_R} \boldsymbol{\epsilon}(\mathbf{w}) : \mathfrak{C}\boldsymbol{\epsilon}(\mathbf{u}) d\boldsymbol{\xi} = \iiint\limits_{\Omega_R} \mathbf{w} \cdot \mathbf{b}_R d\boldsymbol{\xi}. \tag{3.3}$$

Given this final weak form, discrete subspaces $\mathbf{U}^h \subset \mathbf{U}$ and $\mathbf{W}^h \subset \mathbf{W}$ can be chosen in order to complete the discretization. In this work, a basis using linear Lagrange polynomials with compact support on tetrahedral elements was chosen to span the discrete subspaces. Then, a shape function $N_i(\boldsymbol{\xi})$ corresponding to the ith vertex of the tetrahedral mesh is defined such that

$$\mathbf{u} \approx \mathbf{u}^h = \sum_{i \in \mathcal{V}} N_i(\boldsymbol{\xi})\mathbf{u}_i \qquad \text{and} \qquad \mathbf{w} \approx \mathbf{w}^h = \sum_{i \in \mathcal{V}} N_i(\boldsymbol{\xi})\mathbf{w}_i.$$

For a single tetrahedral element e with local displacements/test coefficients $\mathbf{u}_i^e/\mathbf{w}_i^e$ the sum can be rewritten in matrix form, *i.e.*,

$$\mathbf{u}^{h,e} = \mathbf{N}^{e,3}(\boldsymbol{\xi}) \begin{pmatrix} \mathbf{u}_1^e \\ \mathbf{u}_2^e \\ \mathbf{u}_3^e \\ \mathbf{u}_4^e \end{pmatrix} \qquad \text{and} \qquad \mathbf{w}^{h,e} = \mathbf{N}^{e,3}(\boldsymbol{\xi}) \begin{pmatrix} \mathbf{w}_1^e \\ \mathbf{w}_2^e \\ \mathbf{w}_3^e \\ \mathbf{w}_4^e \end{pmatrix}$$

$$\mathbf{N}^{e,m}(\boldsymbol{\xi}) = \begin{pmatrix} N_1^e(\boldsymbol{\xi})\mathbb{1}^m & N_2^e(\boldsymbol{\xi})\mathbb{1}^m & N_3^e(\boldsymbol{\xi})\mathbb{1}^m & N_4^e(\boldsymbol{\xi})\mathbb{1}^m, \end{pmatrix}$$

where $\mathbb{1}^m$ denotes the m-dimensional identity matrix. The four local indices can then be mapped to the M global indices corresponding to the vertex numbering in the tetrahedral mesh using constant selector matrices $\mathbf{S}^e = \delta_{g,g^e(l)} \in \{0,1\}^{M\times 4}$. Here g denotes a global and l a local coefficient index while $g^e(l)$ denotes a function that maps the local index l of an element e to the global index.

Discretizing the weak form (3.3) with the discrete spaces $\mathbf{U}^h$ and $\mathbf{W}^h$, dropping the test coefficient vector, and extending the model using a damping term yields the system of ODEs

$$\mathbf{M}\ddot{\bar{\mathbf{u}}} + \mathbf{D}\dot{\bar{\mathbf{u}}} + \mathbf{K}\bar{\mathbf{u}} = \mathbf{F}^{\text{ext}}$$

$$\mathbf{M} := \sum_e \mathbf{S}^e \iiint\limits_{\Delta^e} \rho_R {\mathbf{N}^{e,3}}^T \mathbf{N}^{e,3} d\boldsymbol{\xi}\ (\mathbf{S}^e)^T, \quad \mathbf{K} := \sum_e \mathbf{S}^e \iiint\limits_{\Delta^e} \mathbf{B}^{eT}\, \mathfrak{C}\, \mathbf{B}^e d\boldsymbol{\xi}\ (\mathbf{S}^e)^T$$

$$\mathbf{F}^{\text{ext}} := \sum_e \mathbf{S}^e \iiint\limits_{\Delta^e} \mathbf{N}^{e,3}\mathbf{b}_R d\boldsymbol{\xi}, \quad B_{ikj}^e = \frac{1}{2}\left(\frac{\partial N_{ij}^{e,3}}{\xi_k} + \frac{\partial N_{kj}^{e,3}}{\xi_i}\right),$$

where $\bar{\mathbf{u}}$ denotes the global vector of nodal displacements (coefficient vector), $\mathbf{M}$ represents the mass matrix, $\mathbf{D} = \alpha\mathbf{M} + \beta\mathbf{K}$ represents the damping matrix following the Rayleigh model, $\mathbf{K}$ denotes the stiffness matrix, and $\mathbf{F}^{\text{ext}}$ the discrete external force vector. The transpose operator applied on the order three tensor $\mathbf{B}^{\mathbf{e}T}$ should be understood such that $B^e_{ikj} = {B^e_{jki}}^T$. Further, Δ^e denotes the domain of the tetrahedron associated with the eth element and $\alpha > 0$ and $\beta > 0$ denote the Rayleigh damping coefficients with which the influence of the mass and stiffness matrix on the dissipative behavior can be controlled. Please note, that the integrand for the evaluation of $\mathbf{K}$ can be simplified when the higher-order tensor $\mathfrak{C}$ and $\mathbf{B}^e$ are rewritten in Voigt notation in order to exploit their symmetry properties. This simplified notation is left out for the sake of brevity but can be found in several works, *e.g.*, M. MÜLLER and M. GROSS [2004]. All occuring integrals have continuous polynomial integrands and can therefore be integrated analytically for a reference element and be transformed to match the integral over the particular tetrahedral domain Δ^e by multiplying the result with the (constant) functional determinant. Further, it is common practice to replace the consistent mass matrix $\mathbf{M}$ by a (diagonal) lumped mass matrix. The lumped representation is acquired by replacing the diagonal entry with the sum of the entries in the respective row and by setting all off-diagonal entries to zero. The ODE system can then be discretized in time using standard time integration schemes. However, the ODE systems resulting from linear elasticity often prove to be very stiff. For that reason the implicit backward Euler method with fixed time step width was employed for time discretization. In order to perform a time step the solution of a linear equation system has to be obtained. Since the matrix is symmetric and positive definite and since the linear equation system is constant as long as the mesh is not modified, the matrix can be decomposed using a Cholesky decomposition once. Multiple steps of the system can then be solved using forward and backward substitution.

Stress and Stress-Rate Recovery

As discussed in the last section the finite element analysis yields a solution of the displacement field over time. Based on this numerical solution the stress field $\mathbf{P}$ and the stress-rate field $\dot{\mathbf{P}}$ can be computed using the constitutive model from the IBVP (3.1). Due to the piecewise linear displacement approximation the stress field is constant on each finite element and therefore exhibits finite jumps between neighboring elements. This is an issue in the subsequent stress analysis as the field is required to be continuous on the whole domain Ω_R to robustly find local stress maxima. To circumvent the problem a (piecewise linear) continuous stress field with nodal coefficient vectors $\overline{\mathbf{P}}_{kl}$ can be recovered by performing an L^2 projection of the fields onto the linear Lagrange FE basis resulting in a piecewise linear, continuous stress field $\mathbf{P}^* = \sum_i N_i(\boldsymbol{\xi})\overline{P}_{kl,i}$. Here, $\overline{\mathbf{P}}_{kl}$ denotes the coefficient vector of the stress tensor field's entry in the kth row and lth column. The projection requires solving a linear equation system for each of the six independent components in the stress and stress-rate tensors, *i.e.*,

$$\mathbf{A}_P\overline{\mathbf{P}}_{kl} = \mathbf{b}_P \quad \text{with } \mathbf{A}_P = \sum_e \mathbf{S}^e \iiint\limits_{\Delta^e} {\mathbf{N}^{e,1}}^T \mathbf{N}^{e,1} d\boldsymbol{\xi}\, (\mathbf{S}^e)^T = \frac{1}{\rho_R}\mathbf{M} \quad \text{and } \mathbf{b}_P = \sum_e \mathbf{S}^e \iiint\limits_{\Delta^e} {\mathbf{N}^{e,1}}^T P^e_{kl} d\boldsymbol{\xi},$$

where P^e_{kl} denotes the (constant) component in the kth row and lth column of stress tensor field according to the eth element. The linear system can be further simplified by lumping the mass matrix-like projection matrix $\mathbf{A}_P$. When the right-hand side is further evaluated analytically the nodal coefficients can be directly determined using

$$\overline{P}_{kl,i} \approx \frac{\rho_R}{4m_{ii}} \sum_{e\in\mathcal{I}} V_e P^e_{kl},$$

where m_{ii} denotes the ith diagonal component of the mass matrix $\mathbf{M}$, $\mathcal{I}$ denotes the set of elements incident to the ith vertex, and V_e denotes the volume of the eth tetrahedral element. The stress-rate field $\dot{\mathbf{P}}$ can then be projected onto the continuous linear basis in the same way. As the fracture criterion will be based on the largest tensile stress component, the stress tensor is decomposed using an eigenvalue decomposition, *i.e.*, $\mathbf{P} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^T$. The time derivative of the principal stress component can further be approximated by $\dot{\mathbf{\Sigma}} \approx \mathrm{diag}(\mathbf{V}^T\dot{\mathbf{P}}\mathbf{V})$ when it is assumed that the eigenvectors and therefore the direction of the principal components changes slowly.

Refinement Pass

The refinement algorithm works best when the mesh is given in a uniformly coarse resolution. The static refinement scheme is then applied locally using a fixed number of refinement steps according to the scheme presented in Section 3.4. More specifically, the field of the locally dominating tensile stress component $\tau = \max(\mathbf{\Sigma})$ is computed to determine the regions with stresses exceeding the material strength τ_{crit} according to the Rankine condition (*cf.* Equation (3.4)). Then, the maximum occurring tensile stress $\tau_{\max}$ of this field is determined per region. Subsequently, all cells incident to a node i lying in a stressed region that fulfills $\tau > \gamma\tau_{\max}$ with $0 < \gamma < 1$ are refined. In all experiments γ was set to 0.8. Finally, the stress analysis is repeated in an iterative fashion to improve the accuracy in highly stressed regions where stress concentrations are expected.

Non-Maximum Suppression

When the iterations in the refinement pass terminate the stress field is analyzed in order to determine if cracks emerge. If this is the case, it is very likely that τ_{crit} is exceeded at multiple nodes in the mesh. Initiating cracks at every node would lead to excessive shattering in the stressed regions. Moreover, the number of initiated cracks is then highly dependent on the mesh resolution. To avoid the shattering it can be assumed that only a single crack emerges at the local maximum of the stress concentration. Therefore, a flood-fill algorithm is used to determine the region of interest where only vertices experiencing a tensile principle stress of $\tau > \gamma\tau_{\max}$ are included. The remaining vertices then serve to indicate a boundary of the flood-filled region. This strategy avoids region-wise shattering while it still allows simultaneous crack initiations caused by several individual stress concentrations. Finally, the procedure yields a relatively small set of crack initiating vertices leading a plausible number of cracks and to a certain independence of the discretization.

Fracture Criterion

As mentioned earlier the Rankine condition is employed to indicate material failure. It states that a brittle material fails if the tensile stress exceeds the material strength, *i.e.*,

$$\tau > \tau_{\text{crit}}. \tag{3.4}$$

Using the non-maximum suppression technique described in the previous section, shattering of highly stressed regions can be avoided. Additionally, the high tensile stresses in the near-field of an emerging crack require a certain time to decay. Therefore, evaluating the Rankine condition might falsely indicate repeated material failure in the same region until the tensile stress has vanished. This artifact will be referred to as temporal shattering. In order to avoid these additional shattering artifacts the

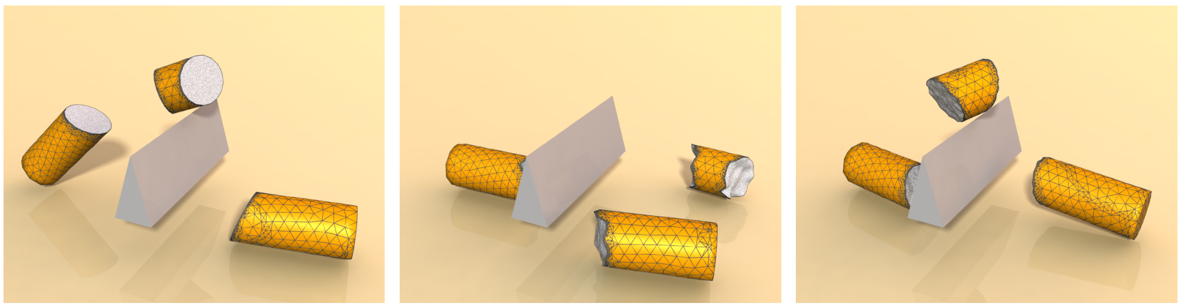


Figure 3.1: A cylinder is fractured on a sharp obstacle using different fracture surfaces. Left: Planar cracks. Center/Right: Planar cracks enriched using a noise functions.

fracture criterion is extended by a second condition:

$$\dot{\tau} \geq \vartheta,$$

where $\vartheta \geq 0$ denotes the critical decrease stress-rate. Physically, this criterion ensures that material portions experiencing a rapid decay in the largest tensile principal stress component do not cause additional cracks.

Fracture Surface Generation

Due to the projection of the piecewise constant stress field onto a piecewise linear Lagrange basis, local stress maxima will always be located at mesh nodes. Therefore, a crack will always originate at a mesh vertex. It can be furthermore assumed that the crack will propagate in the plane orthogonal to the direction of the largest tensile stress. This direction is uniquely defined by the eigenvector corresponding to the eigenvalue representing the largest principal stress component. In the course of the crack propagation the fracture surface may, however, deviate from the plane. For the sake of simplicity, it will be assumed in this work that a crack propagates until it reaches the object boundary. Following this strategy is not physically justified but yields adequate results in many scenarios. A possible way to enforce a finite propagation of a crack was proposed by GLONDU *et al.* [2013] where an energy-based fracture stopping criterion was introduced.

In this work the emerging crack surface is represented using an implicit surface $\Phi(\boldsymbol{\xi}) = 0$ with $\Phi : \Omega_R \rightarrow \mathbb{R}$ describing the signed distance to the surface. Usually a simple plane is used for Φ, but the fracture geometry can be visually enriched, *e.g.*, with noise functions (*cf.* Figure 3.1). To dissect the tetrahedral mesh along the surface all intersected tetrahedra are marked in a breadth first fashion, subsequently duplicated, and finally assigned to each of the according crack flanks. The duplication of the tetrahedra causes overlapping volumes that are treated as virtual material portions following the approach of MOLINO *et al.* [2004]. Finally, the signed distance function Φ is sampled at the vertex locations incident to the duplicated tetrahedra for surface reconstruction purposes as described in the visualization paragraph. If a tetrahedron is intersected more than once, each vertex stores the signed distance with the minimum absolute value.

Coarsening Pass

The employed static mesh adaption scheme is able to uniquely reverse previous adaptions. Details will be discussed in Section 3.4. Therefore, all refinements applied during the numerical stress analysis are reversed in order to save computation time in future analyses. However, due to the duplication of tetrahedra and mesh dissection previously refined regions can not be coarsened. For that reason, all elements intersected by the fracture surface are marked as 'locked', meaning that all refinement or coarsening operations involving a topological change of locked elements are omitted. Due to the natural grading of the refinement scheme, this will still lead to good transitions from very fine structures near crack surfaces to a coarse resolution in the remaining regions (*cf.* Section 3.5).

Separation of Disjoint Partitions

The mesh is separated into multiple meshes if topologically disjoint regions exist where each disjoint region is identified using a flood-fill algorithm. The separation is absolutely necessary, since disjoint regions would be mistakenly treated as a single rigid body in the further simulation process. Additionally, the separation implicitly decomposes the system of ordinary differential equations into smaller equation systems which improves computational performance in further analyses. Finally, all physical quantities required by the rigid body simulator such as the object's mass, inertia, center of mass, *etc.* are updated.

Visualization

To visualize an unfractured object, the boundary of the tetrahedral mesh is extracted. When the object fractures, the boundary triangles from the fractured tetrahedra are removed and the fracture surface is reconstructed using the marching tetrahedra algorithm based on the signed distance field sampled at the vertices of the cut tetrahedra. Note that the signed distance field of multiply cut tetrahedra cannot exactly represent the crack surface such that visual artifacts may arise in form of material loss. As a possible strategy to avoid these artifacts, the approach of SIFAKIS *et al.* [2007] can be considered.

3.4 Static Reversible Refinement Scheme

In this section, a novel reversible refinement scheme for unstructured tetrahedral meshes will be presented. The scheme is designed to gradually refine the mesh and to automatically adapt neighboring elements to avoid T-junctions and non-manifold vertices and edges. The algorithm expects a coarse input mesh that is preferably as uniform as possible in resolution. The basic idea of the novel scheme is that each tetrahedron traverses a maximum of five states within a full refinement cycle. Note that it is not necessary to perform only full cycles for refinement or coarsening as all intermediate states ensure a valid manifold topology. This is especially advantageous as it gives the user enhanced control over the granularity of the mesh. The refinement can also be applied recursively to the tetrahedra which already underwent one or more full refinement cycles. The operations, to transition from one state to the next or previous state, solely consist of topological operations such that the scheme can be broken down into a number of split and flip as well as their inverse operations. In order to capture the current state and cycle in the mesh data structure, a generation index g_e is assigned to each tetrahedron Hence, a tetrahedron e is in state $(g_e \bmod 5)$ and in cycle $(g_e \operatorname{div} 5)$. In the following paragraph, all steps to perform a full refinement cycle will be explained in detail. Each of the steps A-E describe the transition between two states (see Figure 3.2). Some transitions require that neighboring tetrahedra have a specific generation. If this requirement is not fulfilled, the neighboring tetrahedra are refined until the required generation is reached. It is worth noting that the neighboring tetrahedron's generation index can only be too small but never too large because if the neighbor was already refined it would have affected the generation of the currently considered tetrahedron. In order to be able to reverse the refinement, a certain amount of data has to be stored per tetrahedron. A state transition of a single tetrahedron generally results in multiple tetrahedra of different states. In the following, a short description of the states of the newly arisen tetrahedra resulting from a transition will be given in curly braces according to each step. It should be mentioned that some information per tetrahedron has to be stored in order to track the refinement process to ensure the reversibility of the refinement. In some cases it is also necessary to keep track of sets of tetrahedra that will be combined into a single tetrahedron in a coarsening step. This, however, does not require to explicitly store the tetrahedra indices but can be encoded by changing the permutation of faces and edges in the tetrahedron, *e.g.,* in a 4-1 merge operation the first face of each of the four tetrahedra may point away from its three neighbors that are about to be combined, *etc.*

Step A (0 $\leftrightarrow$ 1,1,1,1) To refine a state 0 tetrahedron a 1-4 split is performed. The split inserts a new vertex at the tetrahedron's geometric centroid yielding four new elements associated with state 1 (*cf.* Figure 3.2A). The four child tetrahedra are obtained by moving one vertex of the original tetrahedron to the centroid and generating three new tetrahedra. In order to ensure the reversibility of this operation, it is vital to remember which child is the former original tetrahedron. Hence, the child that represents the original tetrahedron is tagged after the split. Moreover, it must be tracked which four tetrahedra have to be combined in the reverse operation. Therefore, the faces of the four arising tetrahedra are permuted such that the first face of each tetrahedron points away from the three neighboring tetrahedra. Obviously, the inverse operation can be performed by adapting the corresponding vertex at the tagged tetrahedron and deleting the remaining three tetrahedra.

After finishing step A it is necessary to distinguish two cases. Each child contains exactly one face of the parent state 0 tetrahedron. If this inherited face lies in the interior of the object then steps B, C and D describe the further process (*cf.* second row of Figure 3.2). Otherwise, the face lies on the mesh boundary and steps B*, C* and D* (*cf.* third row of Figure 3.2) must be performed in the further process.

Step B (1,1 $\leftrightarrow$ 2,2,2) If the inherited face lies in the interior of the object and if both adjacent tetrahedra are in state 1, the elements are refined using a 2-3-flip. This operation is performed by inserting a new edge connecting the two vertices opposite to the inherited face (*cf.* Figure 3.2B). Then, the two original tetrahedra are modified accordingly and a single new element is created. To ensure that the inverse operation can be applied, the new element has to be tagged in order to remember which tetrahedron has to be deleted. Additionally, the edge uniquely shared by the resulting three tetrahedra has to be stored. Note that the inverse operation must guarantee that the untagged tetrahedra are not accidentally swapped.

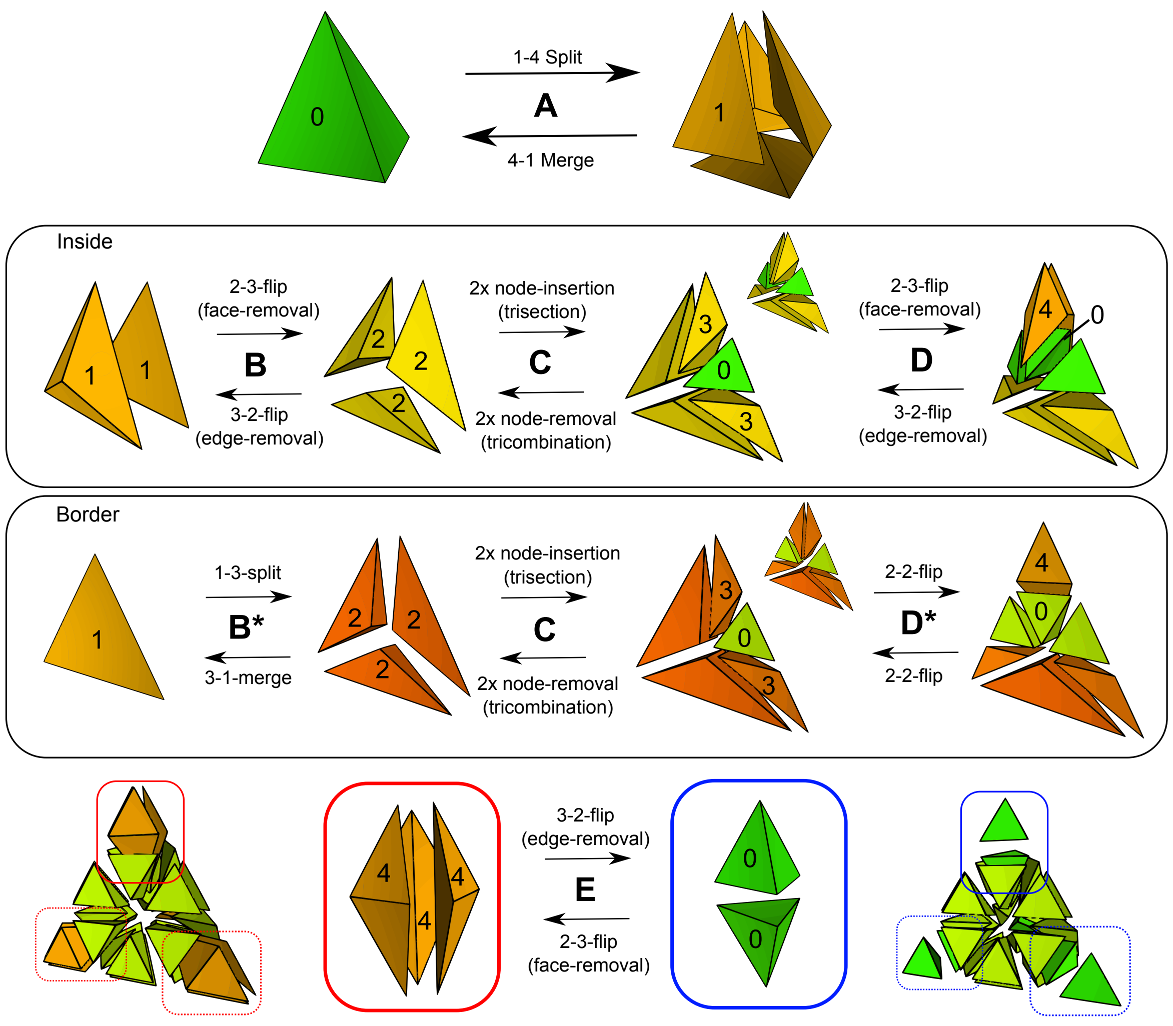


Figure 3.2: Refinement pipeline.

Step C (2 ↔ 0,3,3) The edge opposite to the one inserted and stored in step B is trisected regularly (*cf.* Figure 3.2C). The requirement for this operation is that all tetrahedra incident to this edge share the same generation index. After the trisection, one element in state 0 and two elements in state 3 are generated. State 0 is assigned to one tetrahedron since its quality comes very close to the quality of its original state 0. Therefore, no further operations to complete the cycle are necessary for this element. The state 0 element is created by displacing the vertices of the original tetrahedron and it has to be tagged in order to remember that state 3 and 4 were skipped. Additionally, two new tetrahedra have to be created.

Step D (3,3 ↔ 0,4,4) A 2-3-flip is performed to refine two tetrahedra in state 3 sharing a common face (*cf.* Figure 3.2D). Both tetrahedra are required to have the same generation index. The tetrahedra are then split into three elements by inserting a new edge connecting the vertices opposite to the common face. The operation results in one state 0 and two state 4 elements. A new tetrahedron for the first element is generated and original two tetrahedra are adapted to obtain the remaining two elements. The first element is tagged and must be deleted in the inverse operation.

If the inherited face of a tetrahedron in state 1 lies on the mesh boundary, steps B and D have to be adapted as described in the following.

Step B* (1↔2,2,2) As the inherited face lies on the mesh boundary, there is no adjacent tetrahedron for a 2-3-flip available. To perform a similar refinement step, a vertex is inserted at the geometric centroid of the inherited face (see Figure 3.2B*). This 1-3-split is implemented by adapting the original tetrahedron and by creating two new elements. After the split operation the modified original tetrahedron is marked in order to allow a consistent inverse operation.

Step D* (3,3↔0,4) In the boundary case two tetrahedra in state 3 sharing a face are modified using a 2-2-flip (*cf.* Figure 3.2D*). Both elements are required to have generation index 3. Note that in this case two faces of the elements are coplanar. Therefore, no new elements are generated in this step. The 2-2-flip combines the two tetrahedra and splits the geometry into two new ones using a new edge connecting the vertices opposite to the common face. The new elements are in state 0 and 4, respectively, and can be obtained by adapting the original tetrahedra. The inverse operation only requires to revert the adaption of the elements. No tetrahedra have to be tagged in this step.

The last refinement step does not have to be specialized and is therefore generally applicable to interior and boundary tetrahedra.

Step E (4,4,4 ↔ 0,0) After applying the previous refinement steps B(*)-D(*) to at least three tetrahedra that result from step A, three state 4 elements are located at each corner of the initial state 0 tetrahedron (*cf.* Figure 3.2E, left). The three elements at a corner can then be refined by a 3-2-flip. This flip yields two new tetrahedra in state 0 (see Figure 3.2E, right). This refinement step is realized by adapting two of the three state 4 tetrahedra and deleting the last one. Please note that the deleted tetrahedron contains no information required to inverse any operation, since the tetrahedron was created in the previous step without storing anything. Hence, it can be safely removed. No tetrahedra have to be tagged in this step.

Information Storage

As discussed earlier, a certain amount of information has to be stored to enable the reversion of the refinement. Firstly, a counter storing the generation is required for each tetrahedron. Secondly, some refinement steps require the storage of explicit information to allow consistent inverse operations:

- Step A: 1 bit to tag the parent tetrahedron
- Step B(*): 1 bit to tag the newly added tetrahedron (or parent tetrahedron in case of B(*) + 3 bits to store the index of the common edge
- Step C: 1 bit to tag the parent tetrahedron
- Step D(*): 1 bit to tag the newly added tetrahedron
- Step E: 0 bits.

Consequently, each tetrahedron requires to store 7 bits to ensure the reversibility of the refinement. *E.g.,* using a 32 bit integer per tetrahedron, up to three full refinement cycles (21 bits) with a generation counter (4 bits) can be encoded. Alternatively, 64 bit integer can encode up to 8 full refinement cycles (56 bits) with a 6 bit generation counter.

3.5 Results and Discussion

In this section the proposed approach will be discussed and simulation results will be presented. Firstly, the preservation of mesh quality during local adaptive refinement with the presented static refinement scheme will be examined. Secondly, several complex scenarios for the simulation of brittle fracture will be described and discussed. Thirdly, the adaptive stress refinement approach will be compared to a high-resolution simulation. Finally, the findings are discussed and the outcome is compared with existing approaches.

Quantity	Input mesh	Cycle 1	Cycle 2
#Vertices	106	448	3862
#Tetrahedra	223	1663	17511
Worst min. angle [°]	30	11.36	11.36
Avg. min. angle [°]	43.28	41.54	41.15
Best min. angle [°]	70.52	70.52	70.52
Avg. vertex valence	8.2	10.2	11.8
Max. vertex valence	13	22	23

Table 3.1: Refinement statistics of for beam mesh refinement example (*cf.* Figure 3.3).

Refinement Statistics

To examine the quality preservation of the novel refinement scheme, a tetrahedral beam mesh was locally refined using two subdivision cycles (*cf.* Figure 3.3). Several measurements corresponding to each refinement cycle are summarized in Table 3.1. The minimum dihedral angle was used as a per tetrahedron quality criterion. The greatest minimum angle, and therefore the best possible minimum angle, is $\arctan(2\sqrt{2}) \approx 70.53°$ which is identical to the dihedral angles of a regular tetrahedron. Obviously, the worst possible angle is 0. This experiment shows that the scheme preserves the average minimum angle to a good extent. Moreover, the maximum vertex valence turned out to be almost constant after the first refinement cycle.

Complex Simulation Scenarios

The presented method was applied to simulate several scenarios in order to demonstrate the robustness and visual realism of the approach. The chosen scenarios have a different physical complexity to show the general applicability in many kinds of situations. It should further be mentioned that the tetrahedra selected for refinement in each iteration of the refinement pass are at least refined using one cycle. Adopting this strategy is not mandatory but reasonable in this particular type of application

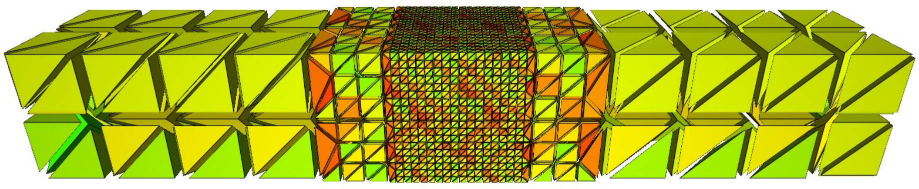


Figure 3.3: Beam refinement test.

because the expensive Cholesky decomposition for time integration has to be recomputed every time the mesh is adapted. This means that more frequent small mesh adaptions would result in a higher computational effort.

In the first scenario the limbs of a brittle Stanford armadillo are fractured by a projectile. While the mesh is initially very coarse (approximately consisting of 3000 tetrahedra), the algorithm produced realistic cracks with high-resolution crack surfaces (*cf.* Figure 3.4, left). In this scenario two full refinement cycles were applied locally in stressed regions. In the second scenario 30 torus-shaped brittle

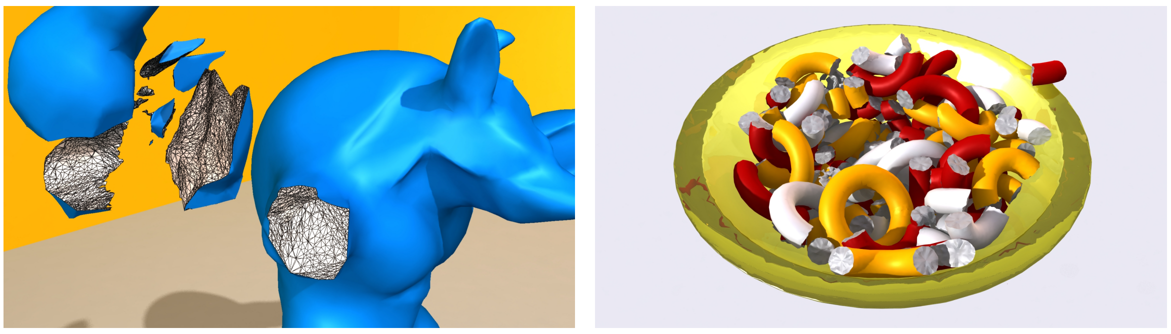


Figure 3.4: Left: A number of projectiles are used to break the limbs of an initially coarse armadillo mesh resulting in high-resolution fracture surfaces. Right: 30 brittle torus-shaped objects are dropped into a static bowl.

objects are dropped into a bowl (*cf.* Figure 3.4, right). The tori fracture due to the collision impact and the bowl collects the resulting fragments. In the third scenario a rigid sphere is used as a projectile to break ten brittle concrete walls (*cf.* Figure 3.5). The tesselation shown in the right image shows how a high-resolution is enforced along the crack surfaces. Both scenes demonstrate the scalability of the approach and that the method yields a realistic and physically plausible behavior even when the objects are subjected to thousands of contacts. In a final scene, a series of three-dimensional extruded letters were dropped from a certain height (*cf.* Figure 3.6).

All results were carried out on an Intel Core i7 860, 2.8 GHz and 8GB RAM. A summary of computation time performance is given in Table 3.2. Besides the matrix factorization and stress recovery algorithm, the biggest portion of the implementation is not parallelized. The most time-consuming parts of the method are the solver initialization, factorization and forward/backward substitution consuming about 50%-70% of computation time, as these steps have to be performed after each topological change in the mesh. In comparison to the solver timings, the fractions of the mesh operations are comparatively small.

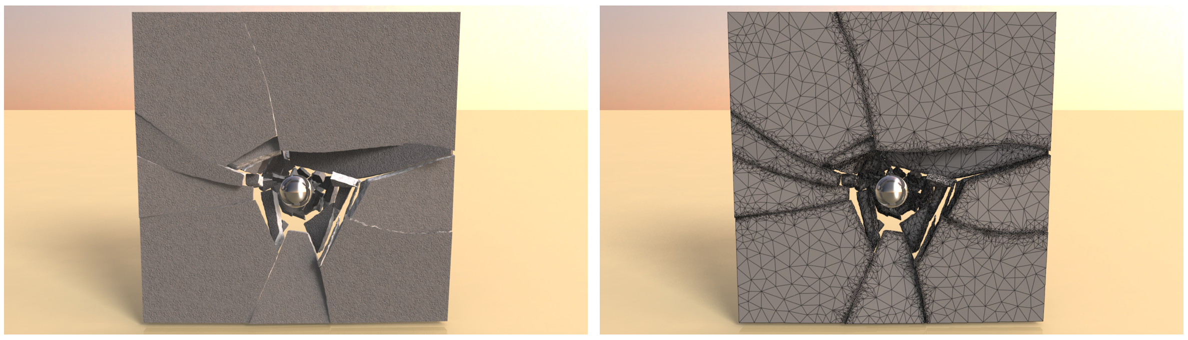


Figure 3.5: Rigid spheres break a series of concrete walls. Left: Final rendered scene. Right. Tesselation resulting from refinement.

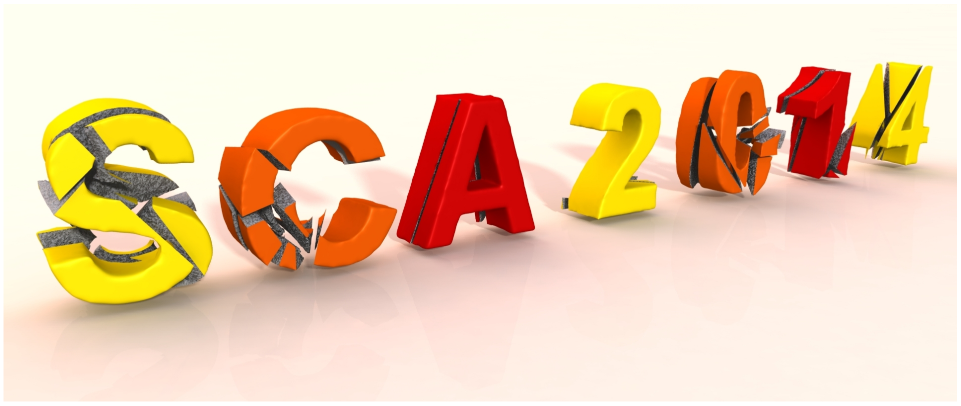


Figure 3.6: Three-dimensional letters are dropped onto the floor and fracture on the impact.

Adaptivity vs. Uniform High Resolution

In order to examine the potential speedup of the adaptive approach over a non-adaptive full high-resolution simulation, a benchmark scenario was simulated in three different configurations. The scenario consisted of a torus dropping on a static floor with the torus' axial symmetry axis being parallel to the floor plane. The three different simulation configurations were chosen as explained in the following. In the first configuration, the scenario was simulated using the described bidirectionally adaptive approach. In the second configuration the adaptive refinement strategy was used but no subsequent coarsening was applied. In contrast to the two adaptive versions, the coarse torus mesh was refined using two refinement cycles prior to the simulation to get a high-resolution mesh and was simulated without any further spatial adaption. The non-adaptive third configuration was the slowest in performance and the unidirectionally adaptive second configuration showed a speedup factor of approximately 15. With the additional coarsening pass applied in the first configuration a speedup factor of 20 was observed.

Discussion

The proposed adaptive approach is able to produce convincing results at moderate cost. The general dynamic behavior of the simulated objects was performed using a rigid body simulator. This simple

Model	**#Cracks**	**Total**	**Avg.**	**Solver**				**Mesh operations**				**#Ref.**
		[s]	[ms]	init.	fact.	subst.	stress	ref.	coar.	frac.	part.	
armadillo	91	22.81	250	25%	24%	17%	23%	2%	2%	6%	1%	10
tori	19636	38.27	1.94	20%	32%	12%	13%	5%	3%	13%	2%	5
walls	18729	76.48	4.08	20%	20%	15%	17%	2%	2%	22%	2%	5
letters	632	4.36	6.89	21%	24%	15%	15%	3%	2%	16%	4%	5

Table 3.2: Performance statistics of simulated scenarios. For each scenario the number of fracture algorithm executions, the total time for the pure fracture computations over the whole domain and the according average computation time per execution is presented. The fraction of suboperations are stated relative to the total time in percent. The solver section summarizes initialization and assembly of the solver matrix, factorization, forward and backward substitution for solving and stress recovery. The next section shows the fractions of mesh operations, more precisely refinement, reverse refinement, fracture generation and separation of disjoint regions. The last column contains the number of refinement steps applied to the underlying meshes during the refinement pass.

strategy is very cost effective and stable compared to full deformable body simulations performed in the works of O'BRIEN and HODGINS [1999] and O'BRIEN *et al.* [2002] where the employed explicit time integration demands very small time steps to ensure a stable simulation. The described FE stress analysis does not require any anchoring of the vertices in contact as used in the method of M. MÜLLER *et al.* [2001] or additional computations to remove null spaces as proposed in the method of BAO *et al.* [2007]. In contrast to the approach of GLONDU *et al.* [2013] the here presented method does not depend on a modal analysis of a static discretization and is moreover able to handle repeated fracturing of fragments while generating high-resolution crack surfaces independent of the initial mesh resolution.

Using the novel refinement scheme, stress analyses of colliding objects were improved. In comparison to red-green refinement [MOLINO *et al.* 2003], the scheme is able to increase the mesh resolution incrementally while the number of tetrahedra is growing slowly. More specifically, one refinement step results in 3 to $3m$ new cells, where m is the number of tetrahedra incident to an edge about to be trisected. The scheme is simple to implement, because it is solely based on simple topological operations. Furthermore, prior refinements can be easily reversed in contrast to the $\sqrt{3}$-refinement based approach of BURKHART *et al.* [2010]. It even preserves sharp features at the boundary geometry which grid-based techniques (*e.g.*, LABELLE and SHEWCHUK [2007] and MOLINO *et al.* [2003]) tend to smoothen.

Naturally, the presented method also comes with limitations. It is most beneficial to use the proposed fracture algorithm with a coarse and uniformly sized initial mesh in order to gain large speedups. However, at the same time this imposes a strong restriction on the amount of surface details a fracture object can have for rendering. In order to serve the different goals of rendering and fracture simulation, the approach could be improved by employing a distinct mesh for each task. M. MÜLLER *et al.* [2004a] proposed a method based on a hexahedral discretization to combine a fracture simulation on objects with an embedded surface meshes. It should be considered to adopt the embedding strategy in the presented work to enhance the visual quality of the simulation. A further limitation is that each arising crack propagates until it reaches the geometric boundary of the simulated object. To account for finite crack propagation, the implementation of the energy-based stopping criterion proposed by GLONDU *et al.* [2013] should be considered.

3.6 Conclusion

In this chapter a novel approach for the adaptive simulation of brittle fracture was presented. The discussed results demonstrate that the algorithm is able to produce realistic animations even when coarse initial discretizations are employed. Due to the adaptive mesh refinement the quality of the performed stress-analyses is enhanced and high-resolution crack surfaces can be produced. Using the non-maximum suppression and the stress-rate based fracture criterion, spatial and temporal shattering artifacts are avoided while the reversion of previous refinements greatly improves the efficiency. By performing dynamic stress analyses, any anchoring of the colliding objects as well as null space eliminations of the solver matrix are avoided. The method can also handle recursive fracturing while the user can control the appearance of crack surfaces by providing modified implicit definitions of the fracture surface. The results demonstrate that the adaptive refinement strategy significantly improves the generation of highly detailed cracks at moderate costs and that the refinement along the fracture surfaces produces appealing crack shapes.

Chapter 4

Robust Cutting Simulation using eXtended Finite Elements

In this chapter a robust remeshing-free algorithm for the simulation of cutting of deformable objects on the basis of the *eXtended Finite Element Method* (XFEM) with fully implicit time integration is presented. The basic idea of the XFEM is to enrich a standard polynomial FE discretization by so-called enrichment functions to improve the quality of the approximation to the underlying PDE. In the particular case of cutting this means that a piecewise constant but discontinuous enrichment function is employed to capture the discontinuity in the displacement field implied by the physical cut. One of the biggest challenges for implementing the XFEM in combination with discontinuous enrichment functions, is the computation of integrals over discontinuous integrands on polyhedral domains. Most existing approaches construct a cut-aligned auxiliary mesh to perform the integration on subdomains where the function is continuous everywhere. In contrast, this chapter presents a cutting algorithm that includes the construction of specialized quadrature rules for each dissected finite element without the requirement to explicitly construct an auxiliary mesh for integration. Moreover, the presented method tackles the problem of avoiding ill-conditioned or even numerically singular solver matrices required for implicit time integration. Therefore, a method to constrain degrees of freedom *Degree of Freedom*s (DOFs) that only marginally contribute to the according entries in the solver matrix is proposed and a novel preconditioner is introduced that efficiently reuses the previously constructed quadrature weights. The method is particularly suitable for the simulation of fine structural cutting as it decouples the added number of DOFs from the cut's geometry and correctly preserves geometric and physical properties by accurately computing the arising element integrals. Owing to the fully implicit time integration, these fine structures can still be simulated robustly using large time steps. As opposed to the enrichment based *eXtended Finite Element* (XFE) strategy, the vast majority of existing methods rely either on remeshing or element duplication to capture the discontinuity imposed by the inserted cuts. An advantage of remeshing based approaches is that they are usually able to correctly preserve physical quantities. However, they strongly couple cut geometry and mesh resolution which can lead to an unnecessary large number of additional DOFs. Element duplication based approaches keep the number of additional DOFs small but fail at correctly conserving mass and stiffness properties. In this chapter, the robustness and physical plausibility of the newly proposed XFEM approach is demonstrated using both simple reproducible academic examples and complex problems with fine structural cutting.

4.1 Introduction

In the last decades the animation of cutting of deformable bodies has been an active research topic and is at the core of applications such as virtual surgery, special effects in feature films or computer games. Maintaining physical plausbility and keeping the simulation robust in scenarios with complex cut paths and fine structured cuts is particularly challenging. It is, moreover, crucial to keep the computational overhead to a minimum even when simulating highly complex scenarios. The vast majority of mesh based approaches for simulating cutting rely on element deletion, element duplication or adaptive remeshing in order to capture a cut's geometry in the simulation mesh. Although, the pure deletion or duplication of elements is easy to implement and leads to a very robust simulation, it does not consistently embed the cut geometry into the discretization leading to physical inconsistencies. Examples for these inconsistencies are unintended loss or addition of mass and stiffness properties. In contrast, proper adaptive remeshing of the dissected regions ensures a physically consistent behavior. This class of approaches, however, adds a large number of additional degrees of freedom (DOFs) to the discrete system in order to properly embed the cut surface into the volumetric simulation mesh. The increased number of DOFs leads then to a substantial overhead in computational cost. This is particularly true if equation systems for implicit time integration have to be solved. Another problem with remeshing based approaches is that all persistent physical quantities such as plastic deformation, temperature *etc.* , that are stored within the discretization, have to be transferred to the new mesh. Besides the additional computation effort associated with the transfer process, this usually leads to undesired diffusion effects as discussed by WICKE *et al.* [2010]. In order to overcome these issues, a novel method for the physical simulation of cutting based on the XFEM with fully implicit time integration is presented in this chapter. The proposed discretization captures cut surfaces with subcell accuracy by means of an appropriate enrichment function without the requirement to apply any topological changes to the simulation mesh. In this way no additional geometry has to be created even when highly complex cut paths are desired. Thus, the computational overhead resulting from geometry processing is kept to a minimum. This results in a simple algorithm where the cut path is encoded via enrichment functions that only have to be considered during the assembly of the equation system for implicit time integration. One of the key problems associated with the XFEM is the evaluation of integrals over discontinuous integrands on polyhedral domains. These are required to compute discretized force vectors as well as the mass and tangent stiffness matrix.

In this chapter an algorithm for the construction of specialized quadrature rules is proposed that approximates the desired integrals with high accuracy. The highly accurate integration scheme enables the simulator to maintain physical properties, *i.e.,* mass and stiffness properties, during cut insertion and propagation. Moreover, an algorithm to constrain nearly non-contributing DOFs and a novel method to precondition the solver matrix by direct use of the constructed quadrature weights are presented. In the results section the method's consistency and advantages over existing approaches are demonstrated using easily reproducible academic examples. Additionally, the robustness of the method is demonstrated in highly complex scenarios with fine-structural cuts.

4.2 Related Work

In this section, methods closely related to the presented approach are discussed. Although the presented method is based on the XFEM which was developed in the field of mechanical engineering, it is targeted towards computer graphics applications. Also some methods for the simulation of fracturing solids are closely related. For that reason the discussion is organized into three paragraphs: approaches for the simulation of cutting and fracture in the field of computer graphics, methods using basis enrichment developed in the field of mechanical engineering, and XFEM based approaches targeted towards computer graphics applications.

Cutting and Fracture in Computer Graphics

In the pioneering work of O'Brien and Hodgins [1999] on brittle fracture, cracks within a solid were captured by explicit embedding of the fracture surface into the underlying tetrahedral discretization on using static local remeshing rules. The concept was later adopted by O'Brien *et al.* [2002] for the animation of ductile fracture. A similar mesh based concept was developed by Bielser *et al.* [1999]. They capture the cut geometry by subdividing a tetrahedral mesh followed by topological disconnection of the resulting subtetrahedra guided by a lookup table. Bielser and M. H. Gross [2000] improved the concept by introducing case specific subdivision rules to reduce the number of additional tetrahedra before the improved approach was generalized in form of a state-machine by Bielser *et al.* [2004]. An alternative static local remeshing approach was proposed by Kaufmann *et al.* [2008] who directly split cut elements along the crack surface but continue the simulation directly by formulating adequate shape functions on the resulting general polyhedra. Moreover, they accurately compute integrals over the smooth shape functions on the polyhedral domains by reducing the volume integrals to integrals over lower-dimensional domains, *i.e.,* element faces and edges.

A particular disadvantage of the mentioned approaches is that the mesh quality can easily deteriorate due to the static remeshing strategy. In order to avoid this deterioration Steinemann *et al.* [2006] decouple visual representation and simulation mesh and dissect the former exclusively on existing faces. Following the same splitting strategy, Yeung *et al.* [2016] proposed a static FEM for linear elastic materials. They further improve the solver efficiency by employing fast rank updates subsequent to matrix augmentations induced by the element splitting. An advantage of this splitting strategy is that it guarantees to fully conserve the mesh quality. However, the approximation can be arbitrarily inaccurate especially when complex cut surfaces are employed. A more elaborate approach was proposed by Wicke *et al.* [2010]. They propose a dynamic local remeshing algorithm based on topological splits and flips as well as vertex smoothing that aims to maintain and improve the mesh quality. A similar two-dimensional approach based on anisotropic remeshing of triangle meshes for the simulation of tearing and cracking of thin sheets was presented by PFAFF *et al.* [2014].

In contrast to embedding the cut surface into a tetrahedral mesh, DICK *et al.* [2010] discretize the underlying elasticity PDE using a regular hexahedral grid. They further excessively refine hexahedra affected by a cut using octree splits and embed the cut by simply separating the fine elements on the axis-aligned faces. An extension of this strategy was later proposed by J. WU *et al.* [2011] where they combine the resulting fine-granular elements to so-called composite finite elements in order to reduce the number of additional degrees of freedom and to keep the solver efficient. MARTIN *et al.* [2008] proposed a method to simulate cutting of deformables using polyhedral finite elements. Starting from a tetrahedral or hexahedral initial mesh, they locally split cut elements into polyhedra and construct harmonic basis function on the arising general polyhedral elements. As opposed to performing local adaptions, an approach for the simulation of multi-layered thin plates using global constrained Delaunay remeshing was proposed by BUSARYEV *et al.* [2013]. However, global remeshing strategies are computationally inefficient as the whole domain has to be remeshed even though only a small portion of the mesh is affected by a cut. Moreover, the change of the discrete basis requires to project the discretized physical quantities onto the new basis. This projection usually suffers from numerical diffusion which was also discussed by WICKE *et al.* [2010].

In order to maintain mesh quality and to decouple cut geometry and mesh resolution, MOLINO *et al.* [2004] developed a VNA. The main idea of the approach is to duplicate cut elements resulting in a potentially non-manifold mesh topology that correctly accounts for the implied discontinuities. The strategy became increasingly popular in the course of the last years and was adopted in several works, *e.g.*, by BAO *et al.* [2007], KOSCHIER *et al.* [2014], and N. MITCHELL *et al.* [2015]. One of the obvious disadvantages of the original approach is that it can not handle an arbitrary number of cuts per element resulting in a maximally-split configuration. Fortunately, by embedding a polygonal auxiliary mesh into the simulation mesh this limitation can be removed as proposed by SIFAKIS *et al.* [2007]. This improved version was later extended by Y. WANG *et al.* [2014] using a spatially adaptive approach that significantly reduces the algorithmic complexity. However, a disadvantage of the cell duplication is that physical properties such as mass and stiffness can not be accurately preserved.

Besides Lagrangian mesh based techniques for cutting and fracture, a meshless approach for fracturing solids using a novel transparency criterion was developed by PAULY *et al.* [2005]. Also a Eulerian mesh based technique was presented by HEGEMANN *et al.* [2013]. They model the discontinuities using level-sets in reference space using a hexahedral background grid.

In contrast to all approaches discussed in this section, the presented method employs a Lagrangian mesh-based discretization but requires no mesh adaptions during runtime. In this way the number of additional DOFs can be kept to a minimum while the discontinuity is captured using the concept of basis enrichment. Moreover, the method is able to correctly maintain discretized physical properties in the course of the simulation as opposed to cell duplication based approaches.

eXtended Finite Elements and Virtual Node Algorithms in Engineering

This paragraph gives a brief overview over related methods developed in the field mechanical engineering. For a general survey on XFEMs for the simulation of solids I would like to refer the reader to the work of FRIES and BELYTSCHKO [2010].

In the pioneering work of Belytschko and Black [1999] the concept of enriched finite element discretizations using discontinuous functions was introduced and later referred to as XFEM. The intended application was the simulation of elastic crack growth in two-dimensional solids by enriching trial and test space using asymptotic displacement fields near the crack tip. Moës *et al.* [1999] built upon this approach and proposed the so-called Heaviside enrichment for the simulation of long cracks using piecewise constant but discontinuous functions. This method was later generalized by Daux *et al.* [2000] for arbitrarily branched and intersecting cracks. A more general approach using harmonic enrichment functions was proposed by Mousavi *et al.* [2011a]. By solving a Laplace problem in the close proximity of the cracked domain they numerically determine the harmonic enrichment function in order to treat multiple intersecting and branched cracks in a unified manner. The approach was later extended for higher-order discretizations by Mousavi *et al.* [2011b].

A method that is strongly related to the XFEM approaches modeling strong discontinuities was proposed by A. Hansbo and P. Hansbo [2004]. The main idea of their approach is to construct independent discrete approximation bases for cut elements on each side fo the cut. Interestingly Areias and Belytschko [2006] showed that the kinematic decomposition of their method is in fact equivalent to the earlier proposed XFEM based version [Moës *et al.* 1999]. During the work of the here presented method, it was moreover discovered that the cell and node duplication following the VNA is identical to the basis modification of A. Hansbo and P. Hansbo [2004]. Based on this insight, it can be concluded that the kinematic decomposition of the VNA and the XFEM is equivalent. However, there is a major difference between the XFEM for strong discontinuities and the traditional version of the VNA. The XFE simulations require to compute integrals over the discontinuously enriched basis functions on each element in the discretization. In the VNA the pure cell duplication implies that these integrals over discontinuous integrands are simply approximated by integrating twice over the same element with the integrands corresponding to each side of the cut.

The mentioned evaluation of integrals over discontinuous integrands poses one of the biggest challenges when implementing the XFEM, as no standard quadrature rules can be applied in order to compute a sufficiently accurate approximation. A naïve approach to solve that problem is to generate a discontinuity-aligned auxiliary mesh on each dissected element and to compute the integrals on the subdomains using Gauss-Legendre quadrature [Moës *et al.* 1999]. However, as one of the main reasons to choose an XFEM based approach is to avoid remeshing, the generation of an auxiliary mesh is undesirable. In this regard, Richardson *et al.* [2011] generate an explicit discontinuity-aligned (lower-dimensional) boundary mesh and reformulate the integrals as surface integrals by applying the divergence theorem. Similar approaches in order to correctly account for the discontinuous integrands in combination with the VNA were developed by the community, *e.g.*, Bedrossian *et al.* [2010], Hellrung *et al.* [2012], Schroeder *et al.* [2014], and Zhu *et al.* [2012]. Over the years, alternative approaches emerged, *i.e.*, adaptive quadrature B. Müller *et al.* [2012], integration over equivalent polynomials Ventura [2006] or the construction of variable quadrature weights Holdych *et al.* [2008]. Due to the fact that adaptive techniques are computationally expensive and as the latter two mentioned methods expect element-wise straight cuts, they are not suitable for cut surfaces of arbitrary shape. In this regard, a novel moment-fitting algorithm for the integration of integrands with discontinuities represented using higher-order level-sets was proposed by B. Müller *et al.* [2013]. The method was later improved by modifying the quadrature node locations B. Müller *et al.* [2017].

The moment-fitting strategy is highly suitable for smooth cut surfaces but suffers from large errors in case of non-smooth, kinked cut shapes.

The approach presented in this chapter represents the cut surface using an explicit triangle mesh. The explicit information is then used to construct specialized quadrature rules for the integration of integrals over discontinuous integrands. Similar to the level-set based approach of B. MÜLLER *et al.* [2013] the quadrature rule is constructed hierarchically. However, the method uses the explicit cut representation to place the quadrature nodes such that integrals on curved and even kinked cut surfaces are accurately captured. Moreover, the approach allows us to place quadrature nodes outside the support domain of the corresponding finite element. In this away the number of required geometric operations and queries is drastically reduced; thus, the implementation is simplified.

eXtended Finite Elements in Computer Graphics

Besides applications in the field of mechanical engineering, the XFEM was also adopted in a few works in computer graphics. An XFE solver for interactive surgery simulation using shifted sign enrichment was proposed by JEŘÁBKOVÁ and KUHLEN [2009]. Their work included a novel mass-lumpling scheme to improve the stability of the simulation in the case of explicit time stepping. KAUFMANN *et al.* [2009] proposed a simulation method for cutting of two-dimensional shells based on an extended discontinuous Galerkin FE discretization and a semi-implicit Euler method for time integration. They introduced the concept of harmonic enrichments that are constructed during runtime by numerically solving a Laplace Problem on the GPU and by storing the resulting solution in a texture. This numerical solution then mimics the behavior of the traditional asymptotic crack enrichment [BELYTSCHKO and BLACK 1999]. This implies that for each cut at least one Laplace problem has to be solved numerically and stored at runtime resulting in a considerable computational effort and memory consumption. Although this strategy proves to be very effective for two-dimensional domains, the computational overhead and memory requirements increase significantly when the approach is generalized to three dimensions. Also the numerical integration over the enriched regions can be computationally expensive as the enrichment is not only discontinuous but also non-polynomial. A strict limitation of this approach is that the detail of the cut geometry is bounded by the texture resolution.

In this chapter a novel XFEM based cutting algorithm featuring stable, fully implicit time integration is presented. An effective preconditioner based on the previously computed quadrature weights is introduced keeping the solver matrix well-conditioned and regular at all times. The later presented results demonstrate that the approach carries out stable simulations even when large time steps are employed.

4.3 Governing Equations and Weak Form

In this section the mathematical formulation for the simulation of deformable objects that are subject to cutting is introduced. The model is based on the continuum model described in Chapter 2 but has to be extended in order to capture and account for the discontinuities in the PDE's solution implied by the cuts. Let us recall and extend the graphic illustration of a continuum in reference and current configuration given in Figure 2.2 for the purpose of cutting simulation (*cf.* Figure 4.1). Besides the usual description of the continuum representing the elastic body, the cut has to be represented. Here, Γ_R

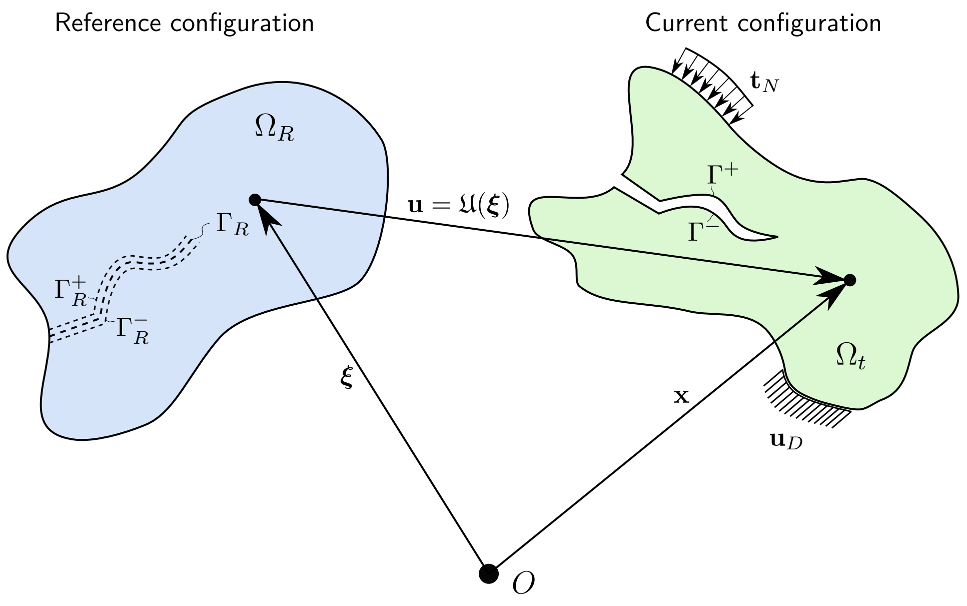


Figure 4.1: Reference and current configuration of a cut continuum. O denotes the origin of a Cartesian coordinate system in Euclidean space.

denotes the cut path in reference configuration with Γ_R^+ and Γ_R^- describing the positive and negative cut flank. Analogously, Γ^+ and Γ^- denote the positive and negative cut flank in the current configuration. Moreover, displacement (Dirichlet) and traction (Neumann) boundary conditions are prescribed on the boundary of the domain. Then, the strong form of the extended IBVP can be derived:

$$\begin{array}{rrcll}
\text{(Linear momentum conservation)} & \rho_R \ddot{\mathbf{u}} & = & \nabla \cdot \mathbf{P} + \mathbf{b}_R & \\
\text{(Constitutive model)} & \mathbf{P} & = & \nabla_{\mathbf{F}} \Psi & \\
\text{(Kinematic equation)} & \mathbf{F} & = & \mathbb{1} + \nabla_{\boldsymbol{\xi}} \mathfrak{U} & \\
\text{(Initial displacement condition)} & \mathbf{u} & = & \mathbf{u}_0 \quad \text{on} & \Omega \text{ at } t = t_0 \\
\text{(Initial velocity condition)} & \mathbf{v} & = & \mathbf{v}_0 \quad \text{on} & \Omega \text{ at } t = t_0 \\
\text{(Dirichlet boundary condition)} & \mathbf{u} & = & \mathbf{u}_D \quad \text{on} & \partial\Omega_D \\
\text{(Neumann boundary condition)} & \mathbf{t} & = & \mathbf{t}_N \quad \text{on} & \partial\Omega_N \cup \Gamma^{\pm}.
\end{array} \tag{4.1}$$

Please note that in this formulation it is assumed that there are no Dirichlet boundary conditions prescribed on the cut flanks. As the most applications where the simulation of cutting is involved require to use large deformations, a non-linear constitutive model should be employed. For all results presented in this chapter the hyperelastic St. Venant-Kirchhoff model was used.

In order to solve the problem numerically using the FEM, its weak form must be derived prior to a polynomial discretization. As already introduced in the previous chapter, the weak form is defined as an inner product of the PDE with a test function that is non-zero on the considered domain. Let $H^1(\Omega_R)$ denote the Sobolev space of square-integrable and once differentiable functions. Then, one can define a vector-valued weak solution space $\mathbf{U}$ for the displacement and a vector-valued weak test space $\mathbf{W}$ such that

$$\begin{aligned}\mathbf{u} \in \mathbf{U} &:= \left\{ \mathbf{u} \in \left(H^1(\Omega_R)\right)^3 \;\middle|\; \mathbf{u}|_{\partial\Omega_D} = \mathbf{u}_D \;\wedge\; \mathbf{u} \text{ discontinuous on } \Gamma_R \right\} \\ \mathbf{w} \in \mathbf{W} &:= \left\{ \mathbf{w} \in \left(H^1(\Omega_R)\right)^3 \;\middle|\; \mathbf{w}|_{\partial\Omega_D} = \mathbf{0} \;\wedge\; \mathbf{w} \text{ discontinuous on } \Gamma_R \right\}.\end{aligned}$$

This definition includes the requirements that the weak solution space strongly fulfills the essential boundary conditions and that the test space vanishes at the Dirichlet boundary. By taking the L^2-inner product of the PDE (*cf.* Equation (4.1)) with the test function on the considered domain, by applying integration by parts and by application of the divergence theorem the weak form is derived as

$$\text{(Weak form)} \quad \iiint\limits_{\Omega_R} \mathbf{w} \cdot (\rho_R \ddot{\mathbf{u}})\, d\boldsymbol{\xi} + \iiint\limits_{\Omega_R} \nabla\mathbf{w} : \mathbf{P} d\boldsymbol{\xi} = \oiint\limits_{\partial\Omega_{R,N} \cup \Gamma_R^{\pm}} \mathbf{w} \cdot \mathbf{t}_N ds + \iiint\limits_{\Omega_R} \mathbf{w} \cdot \mathbf{b}_R d\boldsymbol{\xi}.$$

4.4 Discretization

In this section a standard finite element discretization using linear Lagrange polynomials will be recapitulated. Subsequently, it will be explained how cut surfaces can be geometrically represented and finally the concept of basis enrichment will be introduced that allows one to capture a discontinuity within a FE discretization without remeshing.

Standard FE Discretization

Given the weak form derived in the previous section without accounting for cuts, discrete subspaces $\mathbf{U}^h \subset \mathbf{U}$ and $\mathbf{W}^h \subset \mathbf{W}$ can be chosen in order to discretize the equation. In this work a basis using linear Lagrange polynomials with compact support on tetrahedral elements was chosen to span the discrete subspaces. Then, a shape function $N_i(\boldsymbol{\xi})$ corresponding to the ith vertex of the tetrahedral mesh is defined such that

$$\mathbf{u} \approx \mathbf{u}^h = \sum_{i \in \mathcal{V}} N_i(\boldsymbol{\xi}) \mathbf{u}_i \qquad \text{and} \qquad \mathbf{w} \approx \mathbf{w}^h = \sum_{i \in \mathcal{V}} N_i(\boldsymbol{\xi}) \mathbf{w}_i. \tag{4.2}$$

For a single tetrahedral element e with nodal displacements/test coefficients $\mathbf{u}_i^e/\mathbf{w}_i^e$ the sum can be rewritten in matrix form, *i.e.*,

$$\mathbf{u}^{h,e} = \mathbf{N}^e(\boldsymbol{\xi}) \begin{pmatrix} \mathbf{u}_1^e \\ \mathbf{u}_2^e \\ \mathbf{u}_3^e \\ \mathbf{u}_4^e \end{pmatrix} \qquad \text{and} \qquad \mathbf{w}^{h,e} = \mathbf{N}^e(\boldsymbol{\xi}) \begin{pmatrix} \mathbf{w}_1^e \\ \mathbf{w}_2^e \\ \mathbf{w}_3^e \\ \mathbf{w}_4^e \end{pmatrix} \tag{4.3}$$

$$\mathbf{N}^e(\boldsymbol{\xi}) = \begin{pmatrix} N_1^e(\boldsymbol{\xi})\mathbb{1} & N_2^e(\boldsymbol{\xi})\mathbb{1} & N_3^e(\boldsymbol{\xi})\mathbb{1} & N_4^e(\boldsymbol{\xi})\mathbb{1} \end{pmatrix}.$$

The four local indices can then be mapped to the M global indices corresponding to the vertex numbering in the tetrahedral mesh using a constant selector matrices $\mathbf{S}^e = \delta_{g,g^e(l)} \in \{0,1\}^{M \times 4}$. Here g denotes a global and l a local coefficient index while $g^e(l)$ denotes a function that maps the local index l of an element e to the global index. Semi-discretizing the weak form using Equation (4.2), dropping the test coefficient vector $\overline{\mathbf{w}}$ and extending the module by a damping term following the Rayleigh model then yields

$$\mathbf{M}\ddot{\overline{\mathbf{u}}} + \mathbf{D}\dot{\overline{\mathbf{u}}} + \mathbf{F}^{\text{int}}(\overline{\mathbf{u}}) = \mathbf{F}^{\text{ext}}$$

$$\mathbf{M} := \sum_e \mathbf{S}^e \iiint_{\Delta^e} \rho_R \mathbf{N}^{eT}\mathbf{N}^e d\boldsymbol{\xi}(\mathbf{S}^e)^T, \quad \mathbf{F}^{\text{int}} := \sum_e \mathbf{S}^e \iiint_{\Delta^e} \nabla_{\boldsymbol{\xi}} \left((\mathbf{N}^e)^T\right) : \mathbf{P}(\mathbf{u}^{h,e}) d\boldsymbol{\xi} \tag{4.4}$$

$$\mathbf{F}^{\text{ext}} := \sum_e \mathbf{S}^e \iiint_{\Delta^e} \mathbf{N}^e \mathbf{b}_R d\boldsymbol{\xi},$$

where $\overline{\mathbf{u}}$ denotes the global vector of nodal displacements (coefficient vector), $\mathbf{M}$ represents the mass matrix, $\mathbf{D} = \alpha\mathbf{M} + \beta\mathbf{K}$ represents the damping matrix following the Rayleigh model, where $\mathbf{K} = \nabla_{\overline{\mathbf{u}}}\mathbf{F}^{\text{int}}$ represents the stiffness matrix, and where $\mathbf{F}^{\text{int}}$ and $\mathbf{F}^{\text{ext}}$ denote the discrete internal and external force vectors. Further, Δ^e denotes the domain of the tetrahedron associated with the eth element and $\alpha > 0$ and $\beta > 0$ denote the Rayleigh damping coefficients which steer the influence of the mass and stiffness matrix on the damping behavior.

Cut Representation

As previously mentioned, the described FE discretization is valid as long as the tetrahedral discretization mesh is conforming to the object boundary. This implies that the mesh would have to be adapted in order to capture cuts as long as the discretization does not account for the discontinuity in the solution. In order to capture the discontinuity in the discretization without modifying the mesh topology the discrete trial space is enriched using a step function $\Psi : \Omega_R \to \{-1, 1\}$ as first suggested by MOËS *et al.* [1999] resulting in an XFE discretization. To explicitly define the desired enrichment function a suitable representation for the cut surface geometry has to be found. Especially in the field of mechanical engineering, implicit cut representations, *e.g.*, level-sets defined on grids or on the simulation mesh, are a very popular choice. However, implicit representations are not well-suited to represent sharp features and quickly become very memory intensive. Moreover, it is more than non-trivial to detect if, where and how often an implicit surface has dissected a tetrahedron. For the stated reasons all cut surfaces in this work are represented using explicit triangle meshes in reference space. Consequently, the surface of the jth cut will be denoted Γ_R^j.

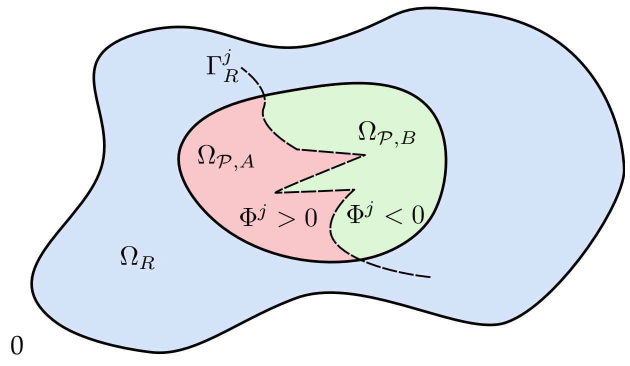


Figure 4.2: Convex domain $\Omega_\mathcal{P}$ that lies within a continuum is dissected by Γ_R^j into exactly two simply connected subdomains $\Omega_{\mathcal{P},A}$ and $\Omega_{\mathcal{P},B}$ such that $\Omega_\mathcal{P} = \Omega_{\mathcal{P},A} \cup \Omega_{\mathcal{P},B}$ and $\overline{\Omega_{\mathcal{P},A}} \cap \overline{\Omega_{\mathcal{P},B}} = \emptyset$.

Assuming that the cut surface Γ_R^j dissects a simply connected, convex domain $\Omega_\mathcal{P} \subseteq \Omega_R$ into exactly two simply connected subdomains $\Omega_{\mathcal{P},A}$ and $\Omega_{\mathcal{P},B}$ (*cf.* Figure 4.2), then the cut surface Γ_R^j implies a level-set function $\Phi^j : \Omega_\mathcal{P} \to \mathbb{R}$ defined as the signed distance from the query point to the surface, *i.e.*,

$$\Phi^j(\boldsymbol{\xi}) = s(\boldsymbol{\xi}) \inf_{\xi^* \in \Gamma_R^j \cup \Omega_\mathcal{P}} \|\boldsymbol{\xi} - \boldsymbol{\xi}^*\| \quad \text{with } s(\boldsymbol{\xi}) = \begin{cases} 1 & \text{if } \boldsymbol{\xi} \in \Omega_{\mathcal{P},A} \\ -1 & \text{otherwise.} \end{cases} \tag{4.5}$$

Basis Enrichment

Using this implicit representation the discrete trial and test spaces (*cf.* Equation (4.2)) can be augmented using an enrichment function in order to capture the discontinuity in the PDE's numerical solution using the XFEM. The main strategy of the (extrinsic) XFEM is to extend an existing (usually polynomial) discrete trial space by additional enrichment functions to capture certain features in the solution, *e.g.*, strong or weak discontinuities, and to improve the quality of the PDE's numerical solution. Mathematically, the discrete trial space is augmented, *i.e.*,

$$\mathbf{u} \approx \mathbf{u}^h = \sum_{i \in \mathcal{V}^0} N_i(\boldsymbol{\xi})\mathbf{u}_i^0 + \sum_{j=1}^{m} \Psi^j(\boldsymbol{\xi}) \sum_{i \in \mathcal{V}^j} N_i(\boldsymbol{\xi})\mathbf{u}_i^j = \sum_{j=0}^{m} \sum_{i \in \mathcal{V}^j} \Psi^j(\boldsymbol{\xi}) N_i(\boldsymbol{\xi})\mathbf{u}_i^j, \tag{4.6}$$

where $m, \mathcal{V}^j, \Psi^j$ and $\mathbf{u}_i^j$ are the number of enrichments, the set of enriched nodes, the enrichment function itself and additional degrees of freedom according to the ith node and jth enrichment, respectively. The first term can further be incorporated in the second sum by defining a trivial enrichment function $\Psi^0(\boldsymbol{\xi}) := 1$ and by setting $\mathcal{V}^0 := \mathcal{V}$, $\mathbf{u}_i^0 := \mathbf{u}_i$. The other additional enrichment functions Ψ^j then have to be chosen such that the discontinuity is captured within the discretization. First proposed by MOËS *et al.* [1999], a piecewise constant step function which is discontinuous along the cut surface can be employed resulting in the so-called *sign* or *generalized Heaviside* enrichment. Building on the level set function defined in Equation (4.5) the sign enrichment function is

$$\Psi^j(\boldsymbol{\xi}) = \operatorname{sgn}(\Phi^j(\boldsymbol{\xi}))\ \forall\, j > 0 \quad \text{with } \operatorname{sgn}(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{otherwise} \end{cases}$$

Unfortunately, defining the enrichment function in this way has several disadvantages. As the enrichment function associated with an enriched node i does not vanish at the node location, the Kronecker-δ property is lost and the numerical solution is not interpolate the values of the degrees of freedom at their locations, *i.e.*, $\mathbf{u}^h(\boldsymbol{\xi}_i) \neq \mathbf{u}_i^0$. This complicates the enforcement of Dirichlet boundary conditions as the nodal displacement coefficients cannot be simply constrained. Fortunately, the Kronecker-δ property can be recovered by shifting the enrichment function at the node locations as proposed by ZI and BELYTSCHKO [2003]. The enrichment function is then specialized for each node and simply shifted by the value of the enrichment function at the node location resulting in

$$\Psi_i^j(\boldsymbol{\xi}) := \frac{1}{2}\left(\Psi^j(\boldsymbol{\xi}) - \Psi^j(\boldsymbol{\xi}_i)\right) \quad \forall\, j > 0,$$

where $\boldsymbol{\xi}_i$ denotes the position of the ith node in the reference configuration. Consequently, Equation (4.6) and the corresponding discrete test space have to be modified accordingly yielding

$$\mathbf{u} \approx \mathbf{u}^h = \sum_{j=0}^{m} \sum_{i \in \mathcal{V}^j} \Psi_i^j(\boldsymbol{\xi}) N_i(\boldsymbol{\xi})\mathbf{u}_i^j \quad \text{and} \quad \mathbf{w} \approx \mathbf{w}^h = \sum_{j=0}^{m} \sum_{i \in \mathcal{V}^j} \Psi_i^j(\boldsymbol{\xi}) N_i(\boldsymbol{\xi})\mathbf{w}_i^j. \tag{4.7}$$

As a result the shifted enrichment function Ψ_i^j vanishes at its associated node i, the linear basis function N_i vanishes at all nodes other than i and a displacement boundary condition at the node can then be enforced by simply constraining the nodal coefficients.

Selecting Nodes for Enrichment

In the last section the XFE discretization for capturing physical cuts in the discrete trial space was introduced. The presented discretization requires to identify a set of nodes $\mathcal{V}^j$ to be enriched for each individual cut. Further, I would like to stress the fact that the correct identification of the set $\mathcal{V}^j$ is crucial because a wrong selection can either lead to the cut flanks physically sticking together or that the discretization has undesired discontinuities in certain regions. Therefore, a strategy to select the correct set of nodes is described in the following.

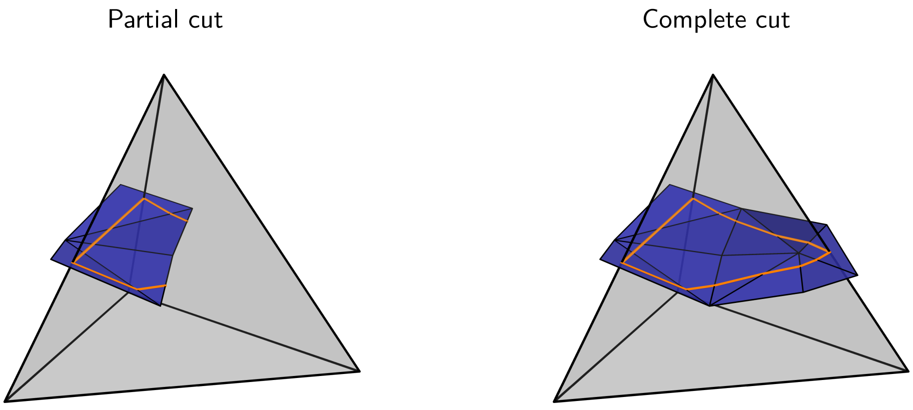


Figure 4.3: Tetrahedral element intersected by cut surface represented by a triangle mesh. The orange polyline represents the intersection path between the surface and the element.

In order to determine $\mathcal{V}^j$, the set of tetrahedra $\mathcal{E}^j$ completely dissected by the jth cut has to be determined. A tetrahedron is further called partially dissected if the intersection path between cut surface and tetrahedron boundary is open and called completely dissected if the intersection path is closed (*cf.* Figure 4.3). Algorithmically, the decision whether a tetrahedron is completely cut is made using the following steps. The algorithm proposed by BARAFF *et al.* [2003] is employed in order to compute the intersection path (*cf.* orange path in Figure 4.3) as the cut surfaces are represented by triangle meshes and because the tetrahedron can be represented by four triangles. Finally, it has to be determined if the cut is complete. This is realized by identifying cycles in the graph implied by the intersection path using a simple depth-first traversal.

Once $\mathcal{E}^j$ is determined, each vertex i incident to a tetrahedron in $\mathcal{E}^j$ is tested whether the support domain of its according shape function N_i is either completely or partially cut. Based on the choice of Lagrange polynomials as shape functions, the support domain of a node i is exactly represented by its one-ring, *i.e.*, the union of incident tetrahedra. Then the boundary mesh of vertex' one-ring represented by a triangle mesh can be extracted and the completeness of the cut can be tested by following the exact same algorithm described for testing the completeness of a cut through a tetrahedron. In case of a complete cut by surface j node i is enriched using the enrichment function Ψ_i^j. Examples for the enrichment test are depicted in Figure 4.4, left using a simplified, two-dimensional example, and in

Figure 4.5 (left) in the three-dimensional case. Special care must be taken when a node's one-ring is cut

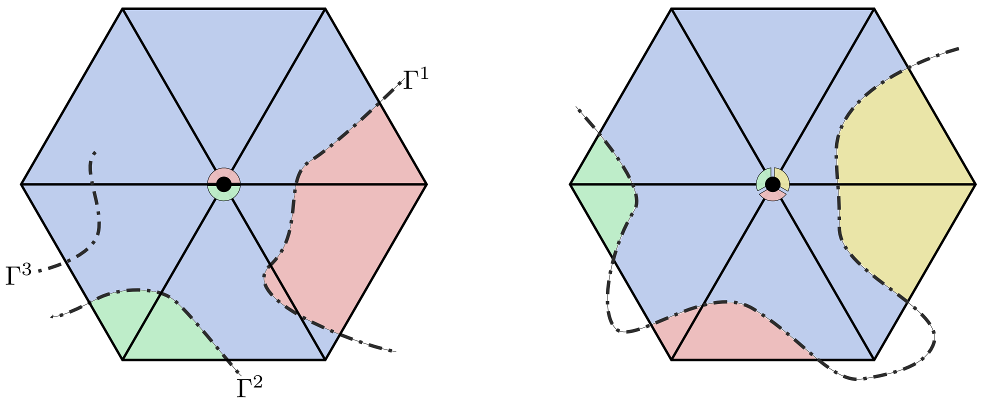


Figure 4.4: Simplified two-dimensional examples for testing whether to enrich a vertex or not. Left: The central node is enriched by Γ^1 and Γ^2, as they completely dissected the one-ring. The partial cut Γ^3 does not lead to an enrichment. Right: One-ring is cut by a single cut path. However, the node is enriched thrice as the one-ring is separated into four disjoint regions.

into multiple disjoint regions by a single cut surface (*cf.* Figure 4.4, right). In order to resolve this case, one can treat each patch consisting of a triangle submesh that completely separates the one-ring as a distinct cut and add an enrichment for each of the patches. Identifying for multiple disjoint regions in three dimensions case has an additional special case as illustrated in Figure 4.5 (right). The cut surface might enter and leave the vertex' one-ring multiple times. However, the one-ring in this example is only dissected into three distinct parts although three closed intersection paths are detected such that the corresponding vertex has to be enriched only twice. This issue can be handled in the following way. The detected paths can then be classified into entering intersection paths (red) and exiting intersection paths (green). Only entering intersection paths cause an enrichment resulting in a correct number of enrichments.

Due to the new extended discrete trial and test spaces defined in Equation (4.7), the element vectors and matrices given in Equation (4.4) have to be modified accordingly. Therefore, a new element-wise discretization similar to the one introduced in Equation (4.3) can formulated, *i.e.*,

$$\mathbf{u}^{h,e} = \mathbf{N}^e(\boldsymbol{\xi}) \begin{pmatrix} \mathbf{u}_1^{e,0} \\ \mathbf{u}_1^{e,1} \\ \vdots \\ \mathbf{u}_1^{e,1} \\ \vdots \\ \mathbf{u}_4^{e,m} \end{pmatrix} \quad \text{and} \quad \mathbf{w}^{h,e} = \mathbf{N}^e(\boldsymbol{\xi}) \begin{pmatrix} \mathbf{w}_1^{e,0} \\ \mathbf{w}_1^{e,1} \\ \vdots \\ \mathbf{w}_1^{e,1} \\ \vdots \\ \mathbf{w}_4^{e,m} \end{pmatrix}$$

$$\mathbf{N}^e(\boldsymbol{\xi}) := \begin{pmatrix} \Psi_1^{e,0}(\boldsymbol{\xi}) N_1^e(\boldsymbol{\xi})\mathbb{1} & \Psi_2^{e,0}(\boldsymbol{\xi}) N_2^e(\boldsymbol{\xi})\mathbb{1} & \dots & \Psi_4^{e,m}(\boldsymbol{\xi}) N_4^e(\boldsymbol{\xi})\mathbb{1} \end{pmatrix},$$

where $\mathbf{u}_i^{e,j}$ and $\Psi_i^{e,j}$ denote the displacement coefficient and shifted enrichment function corresponding to the ith local node of the considered element e and to the jth enrichment. Consequently, with the redefinition of $\mathbf{N}^e$ the definition of the element vectors and matrices given in Equation (4.4) still holds.

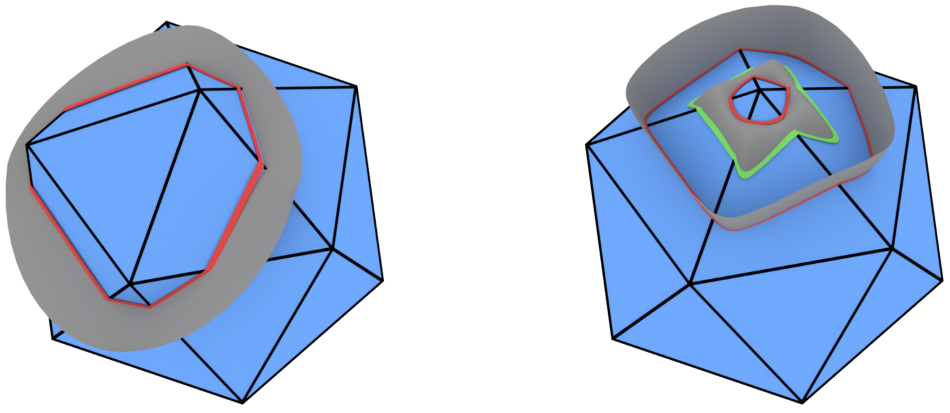


Figure 4.5: One-ring of a vertex is tested for vertex enrichment due to an intersection with the gray cut surface. Left: The vertex' one-ring is exactly separated into two parts by a complete cut at the closed intersection path (red). Right: Cut surface enters and leaves the one-ring multiple times. Although three closed intersection paths are detected (2 × red, 1 × green) the vertex has to be enriched only twice.

Due to the discontinuous nature of Ψ_i^j the integrals required to compute these quantities have discontinuous integrands. Figure 4.6 illustrates a (two-dimensional) triangular finite element separated by a single cut. When integral quantities have to be evaluated on the corresponding domain, special attention has to be paid to the strong discontinuity caused by the enrichment function. This results in two major issues. The first issue is that standard quadrature rules are not applicable anymore which makes the evaluation of these quantities non-trivial. The second issue is that large volume ratios of the resulting subdomains can lead to badly conditioned mass and stiffness matrices.

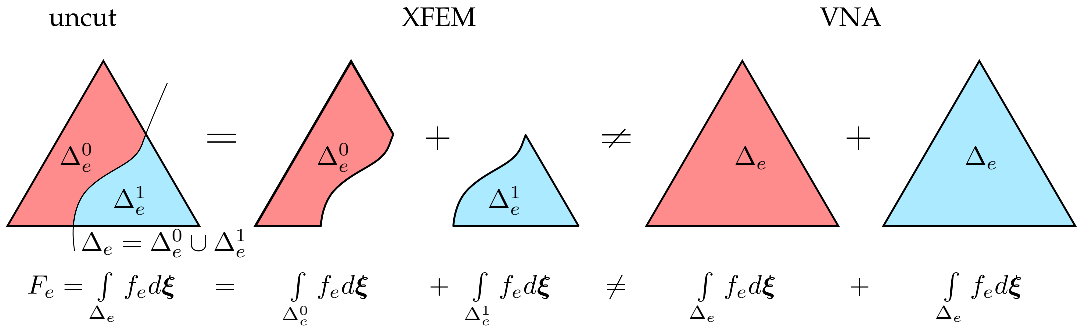

uncut
XFEM
VNA
Δ_e^0
Δ_e^1
$=$
Δ_e^0
$+$
Δ_e^1
$\neq$
Δ_e
$+$
Δ_e
$\Delta_e = \Delta_e^0 \cup \Delta_e^1$
$F_e = \int_{\Delta_e} f_e d\boldsymbol{\xi} = \int_{\Delta_e^0} f_e d\boldsymbol{\xi} + \int_{\Delta_e^1} f_e d\boldsymbol{\xi} \neq \int_{\Delta_e} f_e d\boldsymbol{\xi} + \int_{\Delta_e} f_e d\boldsymbol{\xi}$

Figure 4.6: Integration domains of a dissected element. During integration the arising subdomains left and right of the cut have to be treated separately. While XFEM based methods correctly account for this, VNA methods simply duplicate the element leading to an incorrect integral evaluation.

In the next sections, a novel integration strategy to construct quadrature rules for this specific problem and a preconditioner based on the computed quadrature weights will be introduced. Using this concept both issues will be addressed in a robust manner. At this point I would like to advise the reader with the information that similar cutting methods based on the VNA overcome both the quadrature and matrix condition issues by simply copying the element for each disjoint part that emerged while cutting [MOLINO *et al.* 2004]. While this leads to a very stable simulation, it introduces physical inconsistencies, *i.e.,* mass and stiffness increments as well as a shifted center of mass, which becomes especially noticeable for cutting of fine structures. This effect is also demonstrated in Section 4.7.

4.5 Numerical Integration over Discontinuous Integrands

In order to determine element mass and element tangent stiffness matrices as well as internal and external element force vectors, integrals according to Equation (4.4) have to be evaluated. Based on the standard discretization using polynomial shape functions (*cf.* Equation (3)) all integrands are element-wise continuously differentiable polynomials. An exact evaluation of these can be conveniently achieved by using a Gauss-Legendre quadrature rule of adequate order. However, the extended discretization approach introduced in Equation (4.7) leads to polynomial but discontinuous integrands. Therefore, an evaluation using Gauss-Legendre quadrature is not an option due to the fact that the resulting accuracy is not sufficient. The usage of adaptive techniques yields sufficient results but requires a large number of subdivisions to adequately capture the discontinuity resulting in poor performance [FRIES and BELYTSCHKO 2010]. Another strategy is to remesh the integration domain in order to capture the cut surface and to apply standard Gauss-Legendre quadrature on the resulting subdomains. For one-dimensional integration domains this approach works very well in practice and is very accurate. However, for higher-dimensional domains the remeshing is computationally expensive as it involves the triangulation or the tetrahedralization of possibly non-convex domains and error-prone geometric intersection tests based on floating-point arithmetic.

In the following a method to construct specialized quadrature rules for discontinuous integrands based on the work of B. MÜLLER *et al.* [2013] will be introduced. While a similar approach for quadrature rule construction for partially filled hexahedra was proposed by PATTERSON *et al.* [2012], they require the quadrature points to be positioned inside the filled domain, and rely on computing the integrals of the construction monomials using Monte-Carlo sampling. Also, an approach considering partially filled elements was proposed by KIM and POLLARD [2011]. However, both approaches are not guaranteed to yield an accurate quadrature rule as the sampling of the cell may miss the partially filled regions. In contrast, the here presented approach does not require the points to be inside the considered volume portion and hierarchically constructs rules for integrals over the desired volume portion and the intersecting cut surface. Especially, the rule construction for the evaluation of integrals on the cut surface is beneficial if external tractions have to be applied or if Neumann boundary conditions have to be enforced on the arising surface.

In the following, $\mathcal{T}^d$ denotes the domain enclosed by a d-dimensional simplex. Further, $\overline{\mathcal{T}}^d$ is the closure of $\mathcal{T}^d$, *i.e.,* the union of $\mathcal{T}^d$ and its boundary. It should also be mentioned that, in the following, the double and triple integral notation for integrals over two- and three-dimensional domains, *i.e.,* $\iint$ and $\iiint$, will be deliberately avoided to generalize the notation to the d-dimensional domain $\mathcal{T}^d$. Let $h : \mathcal{T}^d \to \mathbb{R}$ be a function that is only piecewise continuously differentiable on $\mathcal{T}^d$ but continuously differentiable on n_c subdomains $\mathcal{T}_i^d$, where $\overline{\mathcal{T}}^d = \bigcup_i \overline{\mathcal{T}}_i^d$ and $\mathcal{T}_i^d \cap \mathcal{T}_j^d = \emptyset \; \forall \; i \neq j$. Let further χ_i^d be the characteristic function of $\mathcal{T}_i^d$. Then the following integral can be additively decomposed into a sum of individual integrals

$$\int_{\mathcal{T}^d} h(\boldsymbol{\xi})d\boldsymbol{\xi} = \sum_{i=1}^{n_c} \int_{\mathcal{T}^d} \chi_i^d(\boldsymbol{\xi})h(\boldsymbol{\xi})d\boldsymbol{\xi}.$$

One-Dimensional Integration Domain

In order to numerically integrate a piecewise continuous polynomial, the integration domain is subdivided such that the discontinuity is captured and integrated over each individual continuous segment:

$$\int_{\mathcal{T}^1} \chi_i(\boldsymbol{\xi})\, h(\boldsymbol{\xi})d\boldsymbol{\xi} = \int_{\mathcal{T}_i^1} h(\boldsymbol{\xi})d\boldsymbol{\xi}.$$

The position of each discontinuity is determined by a geometrical intersection test between the cut surface and the mesh edge. The integral of each subdomain $\mathcal{T}_i^1$ can then be numerically computed using Gauss-Legendre quadrature. Figure 4.7 shows a one-dimensional example of the discontinuous function and its corresponding subdomains.

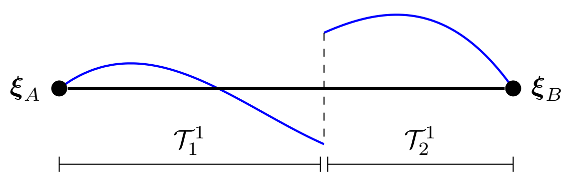


Figure 4.7: Domain of 1D function is subdivided into $\mathcal{T}_1^1$ and $\mathcal{T}_2^1$ such that the function is continuous on both subdomains.

Quadrature Rule Construction for Two and Three-Dimensional Domains

In order to numerically compute integrals over discontinuous integrands on two- or three-dimensional simplicial domains, a specialized quadrature rules depending on the discontinuties' geometries is constructed. More specifically, a single rule for each subdomain $\mathcal{T}_i^d$ where the integrand is continuously differentiable is determined, such that

$$\int_{\mathcal{T}^d} \chi_i(\boldsymbol{\xi})\, h(\boldsymbol{\xi})d\boldsymbol{\xi} \approx \sum_{j=1}^{N} w_{i,j} h(\boldsymbol{\xi}_j), \tag{4.8}$$

where $\boldsymbol{\xi}_j$ and $w_{i,j}$ denote quadrature points and weights corresponding to the ith subdomain, respectively.

Volume/Area Integrals

Given a set of integrands $\mathcal{P} = \{P_1, ..., P_M\}$ over a domain $\mathcal{T}^d$, a quadrature rule can be constructed by solving the following system of equations:

$$\begin{bmatrix} P_1(\boldsymbol{\xi}_1) & \cdots & P_1(\boldsymbol{\xi}_N) \\ \vdots & \ddots & \vdots \\ P_M(\boldsymbol{\xi}_1) & \cdots & P_M(\boldsymbol{\xi}_N) \end{bmatrix} \begin{bmatrix} w_{i,1} \\ \vdots \\ w_{i,N} \end{bmatrix} = \begin{bmatrix} \int_{\mathcal{T}_i^d} P_1 d\boldsymbol{\xi} \\ \vdots \\ \int_{\mathcal{T}_i^d} P_M d\boldsymbol{\xi} \end{bmatrix}. \tag{4.9}$$

As all occurring integrands in Equation (4.4) are polynomials of maximum order two in $\boldsymbol{\xi}$, $\mathcal{P}$ is chosen as a set of (linearly independent) polynomials up to order two. Moreover, choosing $\mathcal{P}$ as a basis that is orthonormal on the reference triangle/tetrahedron greatly improves the matrix condition number of Equation (4.9). The orthonormal basis can be easily constructed from a monomial basis, *i.e.*, $\{1, x, y, x^2, xy, y^2\}$ for $d = 2$, using generalized Gram-Schmidt orthogonalization and subsequent normalization. The equation system is only linear in $w_{i,j}$ but nonlinear in the quadrature nodes $\boldsymbol{\xi}_j$ and for that reason hard to solve. In order to alleviate the problem, a set of quadrature nodes is predefined and their position is kept constant which simplifies Equation (4.9) to a linear equation system.

While the construction of the matrix is trivial given the fixed quadrature points, the evaluation of the system's right-hand-side is challenging due to the integrals over the subdomain $\mathcal{T}_i^d$. In order to tackle this problem the right-hand-side is reformulated by replacing the polynomials P_j with the divergence of their antiderivatives $\mathbf{P}_j^A$, such that $\nabla \cdot \mathbf{P}_j^A = P_j$. Please note, that there are multiple choices to choose the antiderivatives $\mathbf{P}_j^A$. However, $\mathbf{P}_{j,i}^A = \int P_j d\xi_i$ is a valid and simple choice, where $\mathbf{P}_{j,i}^A$ is the ith component of vector $\mathbf{P}_j^A$. Then, the integrals can be rewritten using the divergence theorem yielding

$$\int_{\mathcal{T}_i^d} \nabla_{\boldsymbol{\xi}} \cdot \mathbf{P}_j^A d\boldsymbol{\xi} = \int_{\partial\mathcal{T}_i^d} \mathbf{P}_j^A \cdot \mathbf{n} ds = \underbrace{\int_{\partial\mathcal{T}^d} \chi_i \, \mathbf{P}_j^A \cdot \mathbf{n} ds}_{①} + \underbrace{\int_{\mathcal{I}_i} \mathbf{P}_j^A \cdot \mathbf{n} ds}_{②},$$

where $\mathcal{I}_i = \partial\mathcal{T}_i^d \setminus \partial\mathcal{T}^d$. Here $\mathcal{I}_i$ can be interpreted as the interface between the currently considered subdomain $\mathcal{T}_i^d$ and its neighboring subdomains.

In order to compute integral ① one has to distinguish between the three-dimensional ($d = 3$) and the two-dimensional ($d = 2$) simplicial case. For a tetrahedral domain, again, quadrature rules as described in this section for each triangular face are constructed by computing the intersection of the cut surface's relevant triangles with the plane spanned by the tetrahedron's face. In the two-dimensional case, one can directly compute ① by integrating over the mesh edges as described in the previous subsection.

The second term ② represents a path/surface integral over the interface $\mathcal{I}_i$. For an evaluation of the term an additional interface quadrature rule is constructed as discussed in the next subsection.

Finally, the linear equation system (4.9) can be solved in order to obtain the quadrature weights $w_{i,j}$ required to construct the element vectors and matrices (*cf.* Equation (4.4)). It is important to state, that depending on the number of quadrature nodes and polynomials contained in $\mathcal{P}$ Equation (4.9) is generally over- or underdetermined. For that reason, it is intended to guarantee the equation system to be underdetermined in order minimize the error in a least-squares fashion and to choose the number of quadrature nodes as approximately twice the number of polynomials according to symmetric quadrature rules. Finally, it has to be decided where to place the quadrature points $\boldsymbol{\xi}_j$. For both cases, *i.e.*, $d = 2, 3$, $\boldsymbol{\xi}_j$ can be precomputed according to symmetric quadrature rules with the corresponding number of points following the method proposed by ZHANG *et al.* [2009] and the underdetermined system can be solved by computing the matrix' Moore-Penrose pseudoinverse using a singular value decomposition.

In order to improve the efficiency when solving Equation (4.9) a linear transformation $Q : \mathbb{R}^d \to \mathbb{R}^d$ can be used that maps the simplex onto a reference simplex with coordinates $(0, 0, 0)^T$, $(1, 0, 0)^T$,

$(0,1,0)^T$ for a tetrahedron and $(0,0)^T$, $(1,0)^T$, $(0,1)^T$ for a triangle and the involved cut triangles/segments can be additionally transformed into the reference space. Then the matrix' pseudo-inverse has to be computed only once and can be reused for every element as the matrix is independent of the cut location. Subsequently, the quadrature weights $w_{i,j}$ are scaled by $|\det(Q^{-1})|$ in order to obtain the correct weights in the original space.

Interface Integrals

As an integral over the subdomain interface $\mathcal{I}_i$ has to be computed in order to evaluate term ②, a second quadrature rule with a set of predefined quadrature nodes is constructed such that

$$\int_{\mathcal{I}_i} h ds \approx \sum_{j=1}^{N_i^{\mathcal{I}}} \omega_{i,j} h(\boldsymbol{\xi}_j),$$

where $\boldsymbol{\xi}_j$ and ω_j denote quadrature nodes and weights, respectively.

Based on a vectorial, polynomial basis $\mathcal{P}^D = \left\{\mathbf{P}_1^D, \ldots, \mathbf{P}_{M_D}^D\right\}$, the according construction equations are

$$\begin{bmatrix} \mathbf{P}_1^D(\boldsymbol{\xi}_1)\cdot\mathbf{n}(\boldsymbol{\xi}_1) & \cdots & \mathbf{P}_1^D(\boldsymbol{\xi}_N)\cdot\mathbf{n}(\boldsymbol{\xi}_1) \\ \vdots & \ddots & \vdots \\ \mathbf{P}_M^D(\boldsymbol{\xi}_1)\cdot\mathbf{n}(\boldsymbol{\xi}_1) & \cdots & \mathbf{P}_M^D(\boldsymbol{\xi}_N)\cdot\mathbf{n}(\boldsymbol{\xi}_N) \end{bmatrix} \begin{bmatrix} \omega_{i,1} \\ \vdots \\ \omega_{i,N} \end{bmatrix} = \begin{bmatrix} \int_{\mathcal{I}_i} \mathbf{P}_1^D \cdot \mathbf{n} ds \\ \vdots \\ \int_{\mathcal{I}_i} \mathbf{P}_M^D \cdot \mathbf{n} ds \end{bmatrix}. \tag{4.10}$$

Analogously to Equation (4.9) the evaluation of the linear system's right-hand-side is non-trivial for a choice of arbitrary polynomial vectors. Fortunately, a special choice of $\mathcal{P}^D$ as divergence-free basis allows one to elegantly simplify the right-hand-side. In this work the divergence-free basis was constructed following the method presented by B. MÜLLER *et al.* [2013]. Assuming that $\nabla_{\boldsymbol{\xi}} \cdot \mathbf{P}_j^D = 0$ the entries of Equation (4.10)'s right-hand-side can be transformed using the divergence theorem resulting in

$$\int_{\mathcal{I}_i} \mathbf{P}_j^D \cdot \mathbf{n} ds = \cancelto{0}{\int_{\partial\mathcal{T}^d} \nabla_{\boldsymbol{\xi}} \cdot \mathbf{P}_j^D d\boldsymbol{\xi}} - \int_{\partial\mathcal{T}^d} \chi_i \, \mathbf{P}_j^D \cdot \mathbf{n} ds.$$

The integral can then either be directly evaluated by piecewise integration over the mesh edges for $d = 2$ or using the already constructed quadrature rule used to evaluate ① for $d = 3$.

Again, one has to choose quadrature points $\boldsymbol{\xi}_j$ in order to evaluate the integrands in meaningful positions. For $d = 2$ the cut path is a polyline. Therefore, four quadrature points are placed on each segment following the corresponding quadrature points of traditional Gauss-Legendre quadrature as illustrated in Figure 4.8, left. Due to the dependence of the system matrix on the cut geometry the matrix is not guaranteed to have full rank. The aim is to still maintain an underdetermined system by using enough quadrature points. Therefore, using approximately twice as many points as polynomial basis vectors in $\mathcal{P}^D$ was a safe choice for all of the performed tests and simulations. Consequently, one segment is subdivided at a time until the desired number of quadrature points is reached. Similarly, the quadrature points for $d = 3$ are placed on the triangles of the intersecting patch, again, using the abscissae of the symmetric quadrature rules for triangles determined following the approach of ZHANG *et al.* [2009] (*cf.* Figure 4.8, right).

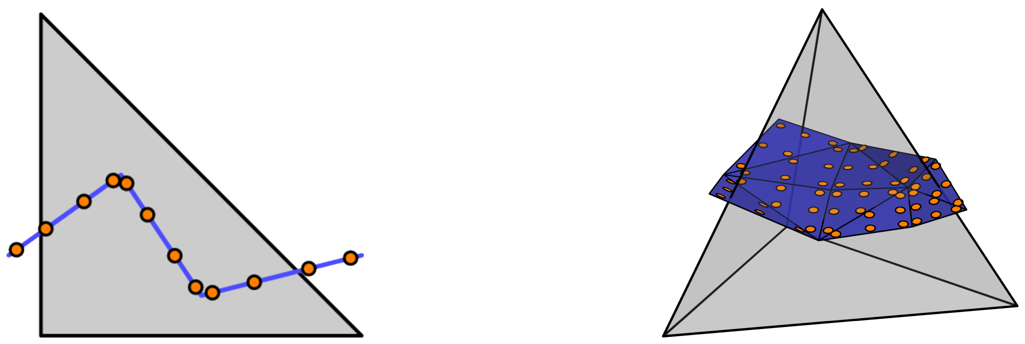


Figure 4.8: Interface quadrature nodes. Blue line/surface represents the cut geometry while orange dots indicate the location of quadrature points. Left: 2D case; triangle intersected by polyline. Right: 3D case; tetrahedron intersected by triangle mesh.

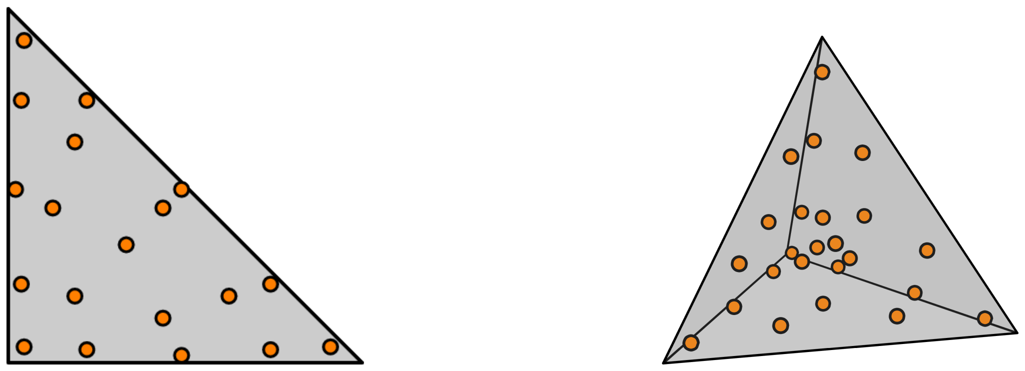


Figure 4.9: Node distribution example of quadrature on 2- and 3-simplex.

Finally, Equation (4.10) can be solved in order to obtain the interface quadrature weights ω_i. Please note, that, again, the number of quadrature points is chose to be approximately twice the number of polynomials in $\mathcal{P}^D$ which results in an undetermined system. Moreover, the system matrix is in general rank-deficient and has poor conditioning depending on the discontinuity's geometry. To robustly solve the underdetermined system, the Moore-Penrose pseudoinverse using a singular value decomposition can be used.

Placing the quadrature nodes directly onto the cut path/surface rather than somewhere within the surrounding finite element [B. MÜLLER *et al.* 2013] ensures that the cut patch's geometry, if smooth or sharp, is captured sufficiently resulting in an accurate quadrature rule. It might, however, seem to be an interesting choice to place the quadrature nodes on the polyline segments/triangles without clipping the primitives on the considered triangular/tetrahedral element. Conducting several experiments showed that even if many quadrature points lay outside the element due to the cut path segments being large compared to the element size, still resulted in very accurate quadrature rules. I assume that this is the case because the integrands, *i.e.*, polynomials, are infinitely often differentiable and therefore very smooth.

4.6 Keeping the Solver Matrix Well-Conditioned

The enrichment of nodes using shifted sign enrichment as described in Section 4.4 adds DOFs to the system and therefore leads to additional entries in the system matrix. The matrix entries corresponding to a DOF heavily depend on how the node's support domain, *i.e.*, its one-ring, is dissected. If the ratio of the contributing subdomain's volume (the portion of cut-off material) and the one-ring's volume is high, the system matrix' condition number deteriorates. This phenomenon is similar to the influence of element shapes on the matrix condition number in standard FE discretizations as discussed by SHEWCHUK [2002].

The convergence of iterative solvers is heavily influenced by the condition number of the system matrices (see *e.g.*, SHEWCHUK [1994]) but also modern direct solvers rely on well-conditioned matrices (see *e.g.*, SONNEVELD [1989]). Even worse, a numerically singular system can cause a breakdown of the simulation. For the stated reasons keeping the system regular and moreover well-conditioned is a vital requirement and one of the key aspects regarding stability and robustness. A well-conditioned system is therefore ensured by means of the following three steps:

1. **Perturbation step** It is tested if the cut surface touches (but does not intersect) the support domain of the currently processed node's shape functions. As previously described, a vertex is enriched if its one-ring is completely dissected. Due to the nature of floating point arithmetic, a complete cut may be detected but no actual intersection geometry is generated. This may cause the construction of the interface quadrature rule to fail, and results in a potentially singular system matrix. Therefore, the cut surface's vertices are perturbed in order to improve robustness. As this problem is caused by numerical inaccuracies in geometric intersection tests, intersection algorithms based on robust predicates or exact arithmetic may be employed to correctly resolve this problem.

2. **Constraining step** If the support domain's volume of an enriched node is small compared to the volume of its shape functions' support domains, the DOF is constrained. In order to avoid the material "sticking" together in the region close to the constrained DOF, the DOF is rigidly moved with its according fragment to keep the afflicted region as-rigid-as-possible. Mathematically, the criterion for constraining a DOF $d_{i,j}$ is

$$\frac{V_{i,j,\text{enr}}}{V_{i,\text{supp}}} < \epsilon,$$

where $V_{i,\text{supp}}$ is the volume of the support domain of the ith node and $V_{i,j,\text{enr}}$ the volume of the DOF $d_{i,j}$'s support domain volume associated with the ith node and the jth enrichment. Further, ϵ represents a scalar threshold that was chosen to be 10^{-9} for all of the later presented results. Generally, the computation of $V_{i,j,\text{enr}}$ is problematic since it should be avoided to explicitly represent the DOF's support domain. However, the previously computed weights can fortunately be reused to determine $V_{i,j,\text{enr}}$ because the volume over the given domain is equal to $\sum_{e\in\mathcal{C}_i} \int\limits_{\Delta_e^j} 1 d\boldsymbol{\xi} = \sum_{e\in\mathcal{C}_i} \sum_{j=1}^{N} w_{e,j}$, where $\mathcal{C}_i$ is the set of elements incident to vertex i.

3. **Preconditioning step** In the third and final step a diagonal preconditioning matrix $\mathbf{T}$ to improve the system matrix $\mathbf{A}$'s condition number for the subsequent Newton iterations is constructed. In

order to keep the matrix symmetric, the diagonal preconditioning matrix $\mathbf{T}$ is applied bilaterally, *i.e.*,

$$\begin{aligned}\mathbf{A}\mathbf{x} &= \mathbf{c}\\ \mathbf{T}^T\mathbf{A}\mathbf{T}\mathbf{y} &= \mathbf{T}^T\mathbf{c}\\ \mathbf{x} &= \mathbf{T}\mathbf{y}.\end{aligned}$$

As previously mentioned the area ratio of $V_{i,\text{supp}}$ and $V_{i,j,\text{enr}}$ has a big influence on the system matrix' condition number. Therefore, diagonal entries of $\mathbf{A}$ are scaled with the inverse ratio such that

$$\begin{aligned}T_{d_{i,j},d_{i,j}} &= \frac{1}{\sqrt{\nu_{i,j}}}\\ \nu_{i,j} = \frac{V_{i,j,\text{enr}}}{V_{i,\text{supp}}} &= \frac{\sum_{e\in\mathcal{C}_i}\sum_{j=1}^{N} w_{e,j}}{V_{i,\text{supp}}}.\end{aligned}$$

4.7 Results and Discussion

In this section, a technique to generate a representation for visualization purposes is described and results and comparisons are presented. All measurements provided in this section were performed on an Intel i7-6700HQ processor with 2.6 GHz, 4 cores and 16GB RAM. The implicit Euler scheme was employed for time integration and the resulting nonlinear equation system was solved using Newton's method in combination with the PARDISO solver implemented in Intel's Math Kernel Library. The number of quadrature points for volume/area quadrature were 24 and 19, respectively (*cf.* Figure 4.9). Analogously, 4 quadrature points per cut path segment for the construction of triangle quadrature rules and 6 quadrature points per cut surface triangle for the construction of tetrahedron quadrature rules were used (*cf.* Figure 4.8). Please note that Dirichlet conditions were enforced following the approach of B. WU *et al.* [2008] in all presented scenarios. Further, I would like to mention that none of the presented simulations includes collision handling.

Visualization

The simulation yields a numerical solution represented by the displacement field $\mathbf{u}^h(\boldsymbol{\xi}, t)$ to the IBVP described in Equation (4.1). Consequently, a suitable representation has to be found in order to visualize the result in an appealing way. At this point I would like to point out that the generation of a visualization can be understood as post-processing step and that the strategy presented in the following is by no means inextricably linked to the described simulation.

In order to represent the (initially uncut) simulation object's boundary surface, the boundary of the tetrahedral discretization is extracted as triangle mesh in reference space. Subsequently, the geometric intersection path between boundary mesh and cut surfaces is computed using the algorithm proposed by BARAFF *et al.* [2003]. The acquired path is then explicitly embedded into both the border mesh and the cut surface meshes by inserting intersection vertices and the according mesh edges followed by a retriangulation. Because the cut surfaces overlap the boundary mesh, all triangles located outside the boundary mesh are clipped away. Each cut surface mesh is then duplicated in order to represents both sides of the cut. Finally, Equation (4.7) is used to compute the world positions for each vertex in the visualization mesh. If the tetrahedral mesh's boundary is too coarse for visualization purposes or if a higher resolution is desired, an additional high-resolution triangle mesh representing the object boundary can be used. The only requirement is that the mesh is entirely contained in the tetrahedral mesh in order to evaluate Equation (4.7) for mapping the vertices to their respective world space positions. Regarding the presented results, a high-resolution boundary mesh was used to generate the results shown in Figures 4.13, 4.12 (right) and 4.14.

Robust XFE Simulation vs. VNA

Three experiments were performed where the novel XFE based method was compared with the widely used VNA. In the first scenario depicted in Figure 4.10 a cube consisting of five tetrahedra was cut at three different locations. Because of the hinge fixation, the resulting slices fold down and oscillate until they reach a resting position. In all simulations performed with the XFEM based method (blue shading) the slices rest in a position where their centers of mass are exactly placed under the fixations. In contrast, the resting positions of the VNA based simulations (red shading) are noticeably displaced resulting in a physically incorrect and even implausible state. In the second experiment depicted in

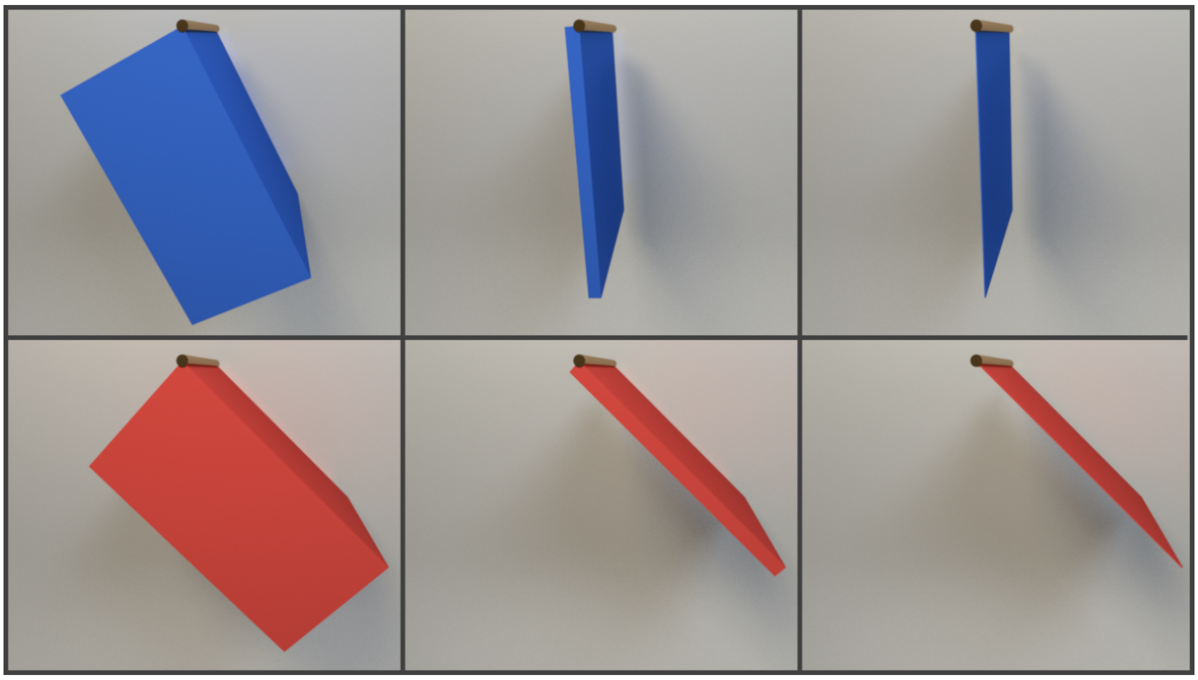


Figure 4.10: Cubes consisting of five tetrahedra fixated on one edge are cut in differently sized slices. This image shows the resting position of the slices where all blue shaded slices where simulated using XFEM based method and the red-shaded slices using the VNA.

Figure 4.11 a deformable slab fixed to a wall was cut into several slices with increasing thickness. The slab is discretized into $61 \times 11 \times 1$ blocks is consisting of five tetrahedra each. The novel XFE approach produces a realistic result where each distinct slice folds down separately due to the varying thicknesses. When the exact same scenario is simulated using the VNA, each slice is as stiff as the uncut slab caused by inexact integration (*cf.* Figure 4.11, right) and therefore scarcely folds. In the

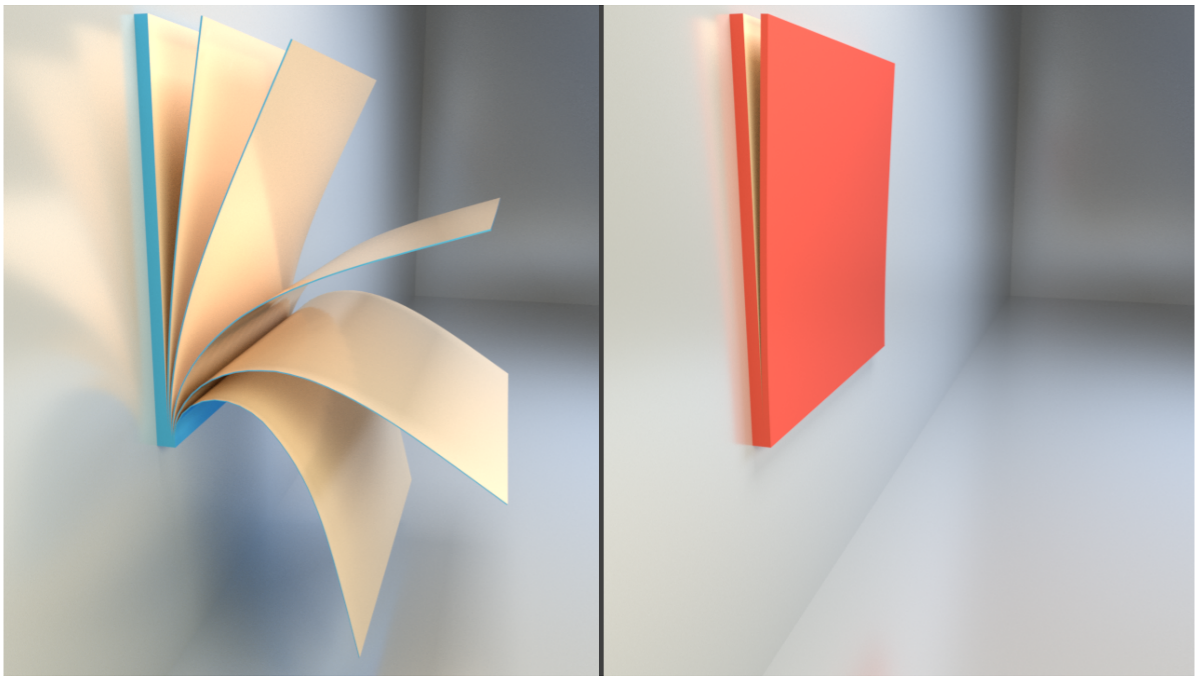


Figure 4.11: A deformable slab is fixed to a wall and cut into five slices with increasing thickness. Left: Resting position resulting from the novel XFE based simulation. Right: Result computed using the VNA with the exact same material and simulation parameters.

third and final experiment mass conservation after the insertion of several cuts was compared. As illustrated in Figure 4.12 (left) two blocks hanging on a comparably small strip of deformable material were simulated. Due to the weight of the blocks the attached strips stretch and can thus be interpreted as nonlinear springs. The nonlinearity in the strips' deformation behavior is caused by the nonlinear hyperelastic constitutive model. As a consequence of the insertion of several cuts into the blocks, the objects unfold. Investigating the unfolded objects' resting states reveals that the simulation performed with the XFE based method conserves mass very well as the attachment point that connects block and fixation strip still rests at the initial position (around one on the background measuring scale). In contrast, the block simulated using the VNA drops significantly, finally resting at three on the background measuring scale. This indicates a significant amount of additional mass caused by the element duplication strategy. Moreover, the resulting dynamic behavior of the object simulated with the novel XFE based method is noticeably "livelier" and looks less stiff compared to the VNA simulation.

Quadrature Comparison

The quality of the acquired volume quadrature rules for 24 quadrature points by evaluating Equation (4.8) on the unit tetrahedron was tested for several test polynomials. In this experiment, three different cut surface shapes were used: a planar, a kinked and a spherical cut. Furthermore, the results were compared with a regular sampling approach as used by KAUFMANN *et al.* [2009] and a specialized adaptive quadrature approach for multidimensional discontinuous functions proposed by

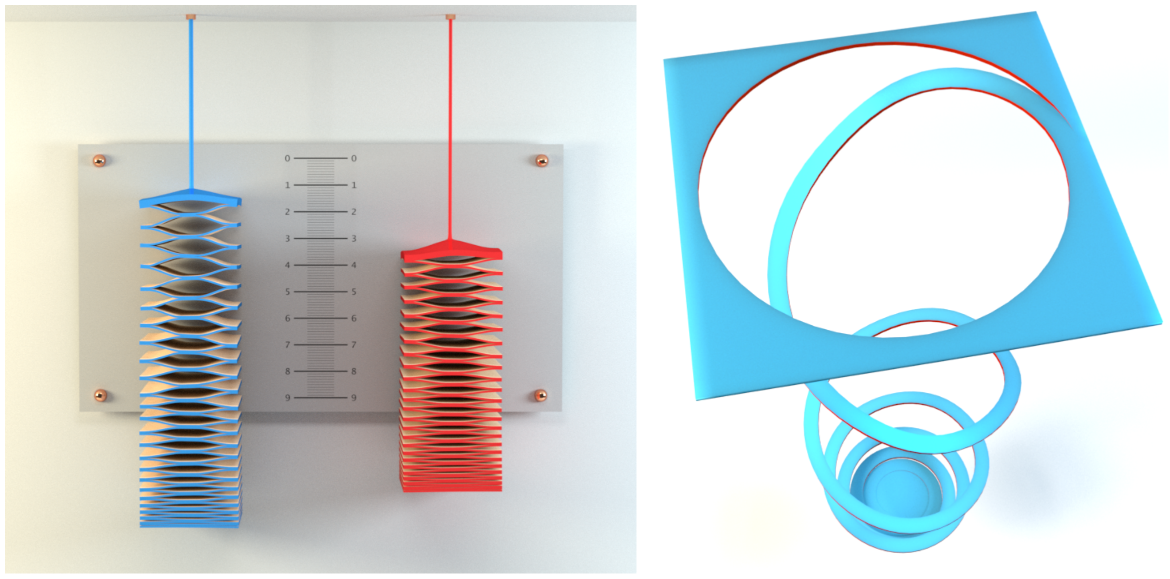


Figure 4.12: Left: Two blocks hanging on thin material strip are cut into an unfolding structure. The result produced by the XFE based method (left) conserves mass before and after inserting cuts as indicated by the strip's extension. The block simulated using the VNA (right) drops significantly due to unphysically added mass and looks stiffer. Right: Cutting a plate with a smooth, helical surface.

B. MÜLLER *et al.* [2012]. To realize the sampling a regular grid on the tetrahedron's bounding box consisting of 50^3 cells was employed. For the adaptive approach a maximum refinement depth of four was used. The results are summarized in Table 4.1. Please note that only quadrature points evaluating to non-zero values were accounted for the values in the #QP columns for the sampling approach and the adaptive quadrature technique. The regular sampling consistently produced less accurate results compared to our method while a much larger number of quadrature points was required. Although the adaptive approach was in some cases able to achieve comparable accuracy, the number of required quadrature points was larger by several orders of magnitude.

Complex Cutting

Besides comparisons and academic examples, four complex simulations were performed. In the first experiment a helical cut surface was used to dissect a plate discretized with $25 \times 25 \times 1$ hexahedral blocks each consisting of 5 tetrahedra (*cf.* Figure 4.12, right). Even though the object is discretized using a small number of linear elements, the helical-shaped material deforms very smoothly while the cut is progressing. In the second scenario the Stanford armadillo that was fixated on its limbs was simulated (*cf.* Figure 4.13, left). This example demonstrate that the presented method is able to robustly handle cuts with low and high frequent smooth and/or sharp features while dissecting thousands of tetrahedra.

In the scenario illustrated in Figure 4.14, a circular groove was cut into the Stanford bunny. Furthermore, the groove itself was peeled off by a second progressing cut. The result demonstrates that the method is able to robustly handle finely structured cut surfaces. Moreover, it shows that the simulation yields realistic results even when the underlying tetrahedral mesh is coarse. The tetrahedral

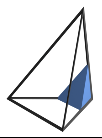
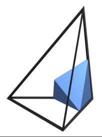
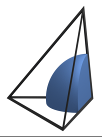
Scenario	Integrand	Pres. Method	Regular Sampling		Adaptive Quadr.	
	$h(\xi, \eta, \zeta)$	ϵ_{rel} (24QP)	ϵ_{rel}	#QP	ϵ_{rel}	#QP
	1	3.32×10^{-5}	4.44×10^{-3}	560	5.79×10^{-2}	54
	$\xi + \eta + \zeta$	4.43×10^{-5}	4.44×10^{-3}	560	8.19×10^{-2}	54
	$\eta\zeta + \xi\zeta + \xi\eta$	5.54×10^{-5}	9.78×10^{-1}	560	9.25×10^{-1}	181
	$\xi^2 + \eta^2 + \zeta^2$	5.54×10^{-5}	5.00×10^{-1}	560	4.94×10^{-1}	181
	1	2.99×10^{-6}	4.94×10^{-4}	3795	1.54×10^{-2}	646
	$\xi + \eta + \zeta$	6.50×10^{-6}	4.89×10^{-4}	3795	2.16×10^{-2}	646
	$\eta\zeta + \xi\zeta + \xi\eta$	1.26×10^{-5}	2.50×10^{-3}	3795	1.86×10^{-2}	2512
	$\xi^2 + \eta^2 + \zeta^2$	1.38×10^{-5}	1.10×10^{-4}	3795	1.72×10^{-2}	2512
	1	3.88×10^{-4}	4.62×10^{-3}	8219	1.21×10^{-2}	788
	$\xi + \eta + \zeta$	5.15×10^{-4}	6.23×10^{-3}	8219	1.65×10^{-2}	788
	$\eta\zeta + \xi\zeta + \xi\eta$	6.43×10^{-4}	7.60×10^{-3}	8219	1.30×10^{-3}	3073
	$\xi^2 + \eta^2 + \zeta^2$	6.46×10^{-4}	7.71×10^{-3}	8219	1.79×10^{-4}	3073

Table 4.1: Results of the numerical integration test using regular sampling, an adaptive approach [B. MÜLLER *et al.* 2012] and the presented approach. The unit tetrahedron was cut using a planar (diagonal) cut, a kinked cut and a spherical cut. The integral over polynomials $h(\xi, \eta, \zeta)$ over the volume portion indicated in blue was computed. ϵ_{rel} and #QP represent the relative error to the analytic solution and the number of the contributing quadrature points used for quadrature, respectively.

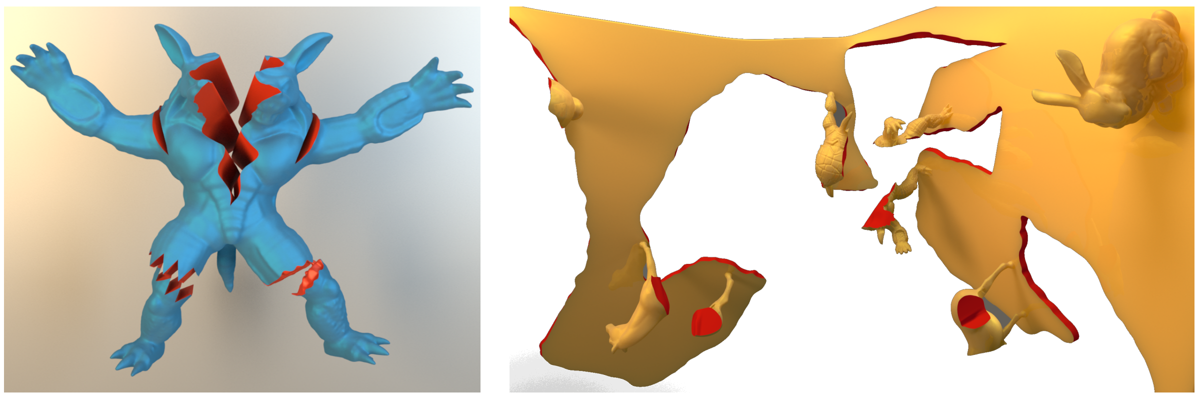


Figure 4.13: Left: Stanford armadillo is cut by several high- and low-frequent, smooth or sharp surfaces. Right: Plate with attached objects is cut by long, bumpy cut surface.

discretization of the bunny consisted of only ~18.5k tetrahedra while the visualized triangle mesh had ~53k triangles. In the fourth and final scenario, a plate with several attached objects as depicted in Figure 4.13 (right) was simulated. It is dissected by a very long complicated cut surface. Due to the complex bumpy structure of the surface several tetrahedra are cut multiple times by the same cut surface resulting in the requirement to enrich several nodes multiple times (problem described in Section 4.4). This highly complex example shows that the presented method is able to accurately handle very complex cut surfaces and yields a realistic animation where all cut regions are properly separating without any artifacts.

Also performance was measured in two scenarios. Both scenarios were simulated using a time step width of $\Delta t = 5$ms. The simulation of the bunny scenario took on average 1.375s per step with initially 5326 DOFs, finally resulting in 9432 DOFs. Further, 47.27% of the time was spent to process the cuts

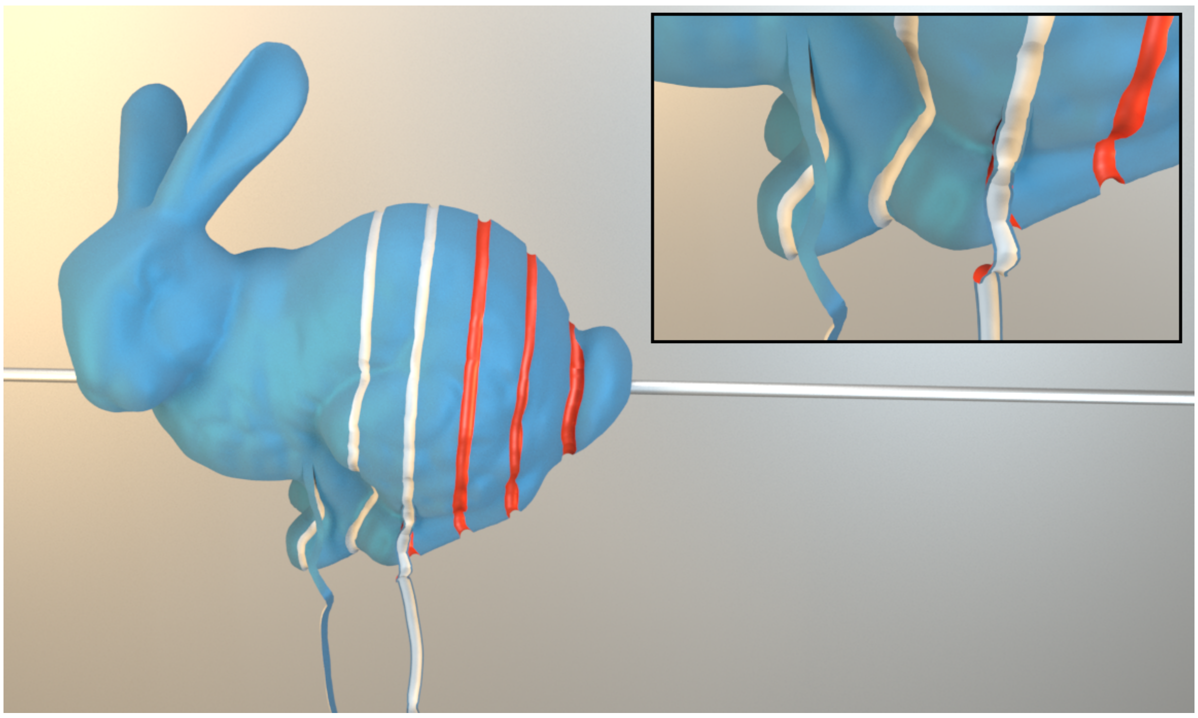


Figure 4.14: A groove is carved into the Stanford bunny (18.5k tetrahedra) rotating around a fixed axis. Subsequently, the groove itself is peeled off by a second cut.

and to enrich the nodes. The quadrature rule construction is included in that portion but individually took only 8.5% of the time required per step. The plate scenario initially consisted of 47643 DOFs resulting in 53326 DOFs and took 9.365s per time step. 17.5% of the time was spent to process the cuts while the quadrature rule construction individually took 0.7% of the simulation time. The last scenario clearly shows the advantage of the presented approach in comparison to remeshing based methods as the number of DOFs increased only by 12% even when a very complex cut surface is employed.

Preconditioning

In order to analyze the effect of the presented preconditioner on the system matrix' condition number, an experiment where a unit cube consisting of five tetrahedra with bounding coordinates $(-1,-1,-1)^T$ and $(1,1,1)^T$ was dissected using a planar cut with surface normal $(1,0,0)$. Then the condition number of the system matrix was measured while varying the location of the cut in x-direction. We further chose the system matrix as $\mathbf{A} = \mathbf{M} + \Delta t^2\mathbf{K}$ resulting from the discretization and linearization in a single Newton step, where Δt and $\mathbf{K} = \partial\mathbf{F}^{\text{int}}/\partial\mathbf{u}$ represent time step width and tangent stiffness matrix, respectively. The cube was unconstrained and the matrix was assembled for Young's modulus $E = 10^6 N/m^2$, Poisson ratio $\nu = 0.3$, density $\rho = 1000 kg/m^3$ and time step width $\Delta t = 10^{-3}s$. The graph in Figure 4.15 shows the outcome of the experiment. While the condition number of the unpreconditioned system matrix approaches infinity when the cut is close to the bounds, the condition number of the preconditioned matrix is considerably lower. Even when the cube is cut exactly through its center, the condition number of the unpreconditioned matrix is nearly two orders of magnitude higher compared to the uncut matrix' condition number. It should further be mentioned that the condition number of the preconditioned matrix will also approach infinity when the cut is so close to the cube boundary that some of the enriched vertices' DOFs have (almost) no support. Fortunately, this is prevented by constraining the affected nodes which keeps the condition number bounded.

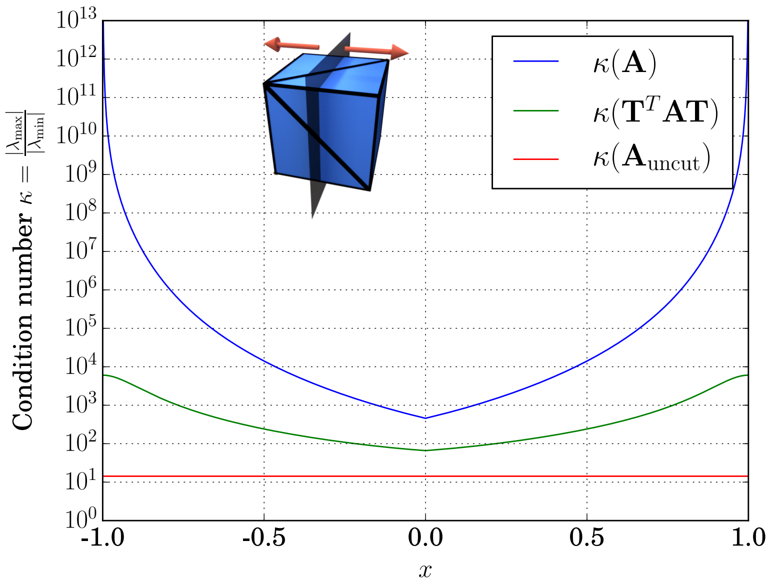


Figure 4.15: Semi-logarithmic plot of the condition number of the system matrix $\mathbf{A} = \mathbf{M} + \Delta t^2\mathbf{K}$ over cut location x. A cube consisting of five tetrahedra was cut using a planar cut surface.

4.8 Conclusion

In this chapter a novel approach for the simulation of complex cutting of three-dimensional deformable solids with fully implicit time integration was presented. After introducing the concept of basis enrichment for the representation of cuts within the finite element discretization, an approach to construct specialized quadrature rules to compute arising integrals over discontinuous elements on polyhedral domains was proposed. Moreover, an algorithm was presented that keeps the equation system required for implicit time integration regular and well-conditioned. The presented results demonstrate that the presented method robustly handles complex cut surfaces in large scenarios. It was also shown that the method is able to realistically simulate fine-structural cuts, even in the case of coarse tetrahedral discretizations. The XFE based approach was further compared to the popular VNA where the considerable advantages concerning preservation of physical plausibility such as mass conservation and correctly maintaining stiffness properties of the XFE based approach was demonstrated.

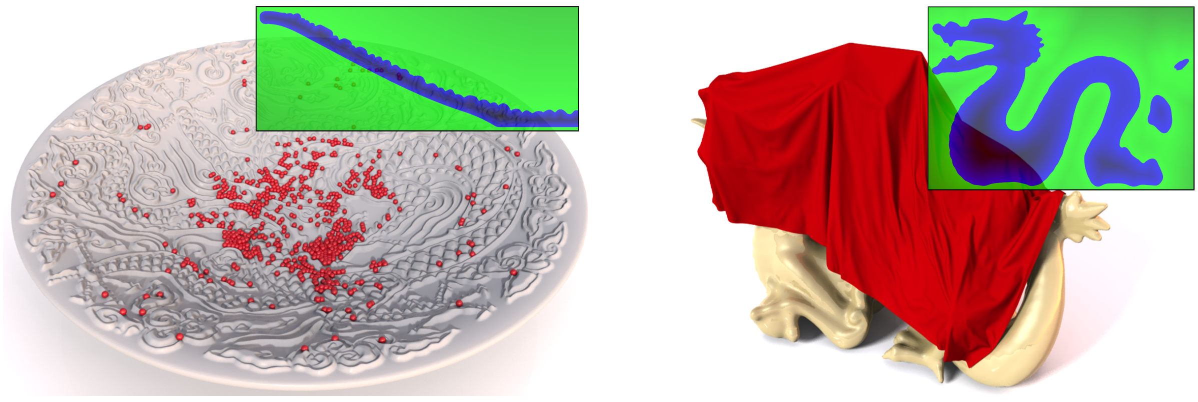
As explained before, the presented method only treats elements as cut if they are completely dissected by the cut surface. Therefore, the approach currently cannot handle cuts that progressively advance within a single element. Building on the flexibility of the XFEM to incorporate different enrichments, one could enrich completely dissected elements using the presented shifted sign enrichment and treat regions near the crack tip following the harmonic enrichment strategy KAUFMANN *et* *al.* [2009]. A localized Laplace enrichment near crack tips would allow one to represent intra-element progressive cuts at moderate cost. This would result in a nice synergy of both approaches. A limitation of the given implementation is that it did not treat the case of mutually intersecting and T-cuts. However, there seems to be no reason why this should limit the generality of the proposed approach as branched and intersecting cuts can probably be directly handled using the specialized enrichments proposed by DAUX *et al.* [2000]. Further, an implicit backward Euler scheme together with Newton iterations was employed to solve the resulting nonlinear equation system. I am aware of the fact that more elaborate and efficient implicit time integration schemes were developed within the computer graphics community in the recent years. While performance was not the main focus of this work, it would still be beneficial to incorporate a more efficient time integrator and to fully parallelize the implementation.

Part *II*

Implicit Boundary and Interface Representations

Chapter 5

hp-Adaptive Generation of Discrete Signed Distance Fields



In this chapter an hp-adaptive algorithm to generate discrete higher-order polynomial *Signed Distance Fields* (SDFs) on axis-aligned hexahedral grids from manifold polygonal input meshes is presented. Using an orthonormal polynomial basis, polynomials are efficiently fit to the underlying signed distance function on each cell. The proposed error-driven construction algorithm is globally adaptive and iteratively refines the SDFs using either spatial subdivision (h-refinement) following an octree scheme or by cell-wise adaption of the polynomial approximation's degree (p-refinement). Furthermore, a novel decision criterion based on an error-estimator is introduced that decides whether to apply p- or h-refinement. The carried out results demonstrate that the method is able to construct more accurate SDFs at significantly lower memory consumption compared to previous approaches. While the cell-wise polynomial approximation will result in highly accurate SDFs, it can not be guaranteed that the piecewise approximation is continuous over cell interfaces. Therefore, an optimization-based post-processing step is proposed that weakly enforces continuity. Finally, the generated SDFs are used in the application of collision detection in the physically based simulation of geometrically highly complex solid objects. This demonstrates the practical relevance and applicability of the proposed representation.

5.1 Introduction

Signed distance fields are a frequently used tool in the field of computer graphics and serve a wide range of applications including surface reconstruction [Calakli and Taubin 2011], rendering [Jamriška 2010], geometrical modeling [S. Frisken 2006] or collision detection [N. Mitchell *et al.* 2015]. For a three-dimensional spatial domain $\mathcal{B} \subset \mathbb{R}^3$ the distance function is usually defined as the Euclidean distance from a given point $\boldsymbol{\xi} = (\xi, \eta, \zeta)^T$ in space to the nearest point on the boundary $\partial\mathcal{B}$ of the domain. The sign of the distance function additionally provides information whether the point in question lies inside or outside the domain. Mathematically, the signed distance function $\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}$ is defined as

$$\Phi(\boldsymbol{\xi}) = s(\boldsymbol{\xi}) \inf_{\boldsymbol{\xi}^* \in \partial\mathcal{B}} \|\boldsymbol{\xi} - \boldsymbol{\xi}^*\|, \quad s(\boldsymbol{\xi}) = \begin{cases} -1 & \boldsymbol{\xi} \in \mathcal{B} \\ 1 & \text{otherwise.} \end{cases} \tag{5.1}$$

Signed distance functions can be evaluated efficiently if an analytic form exists for the associated object. This is the case for simple geometric shapes such as spheres, tori, boxes, *etc*. For arbitrary polyhedral shapes the evaluation of the signed distance function is, however, computationally very expensive. For that reason, it is common practice to discretize the signed distance function in order to evaluate the function more efficiently. In the remainder of this thesis, a discretized signed distance function will be referred to as SDF.

Most commonly, an SDF is constructed by sampling the signed distance at the vertices of a regular hexahedral grid and by trilinearly interpolating within each cell, as *e.g.*, proposed by XU and BARBIČ [2014b]. However, for complex objects this discretization strategy either consumes a large amount of memory or is not sufficiently accurate. The sampling may additionally suffer from aliasing effects. More elaborate approaches sample the function adaptively in order to increase the accuracy in regions with fine details and to reduce the overall memory consumption. The adaptive sampling can be realized, *e.g.*, as an octree-like scheme, as proposed by S. F. FRISKEN *et al.* [2000]. Especially in regions near curved or sharp features, strong subdivision is required resulting in very memory consuming SDF representations.

In this chapter, a novel method to efficiently construct a grid-based SDF using hierarchical *hp*-refinement based on piecewise polynomial fitting is proposed. Besides a spatial adaption using octree subdivision to refine the cell size (h), the polynomial degree (p) of the local discretization is adapted. An orthonormal polynomial basis using shifted, normalized Legendre polynomials is employed that is able to hierarchically construct higher-order polynomials without having to discard and recompute any of the previously computed coefficients. Using a novel *hp*-decision criterion the algorithm estimates whether h- or p-adaption is more beneficial in each individual refinement step. On the basis of several examples for the construction of SDFs for complex surfaces, it will be demonstrated that the method generates highly accurate discretizations while the memory consumption remains at a minimum. Using a novel nearness weighting approach, the user can choose to focus the refinement efforts on regions close to the surface of the associated object. Finally, it is shown in several experiments that the *hp*-adaptive SDFs are well-suited for the robust detection of collisions in dynamic simulations. In addition to the detection of contacts, the SDF also provides information about the penetration depth and contact normals. The method is, however, not limited to this application.

5.2 Related Work

Since the introduction of SDFs by ROSENFELD and PFALTZ [1966] to the computer graphics community, numerous approaches to construct and use SDFs have been proposed. For a general overview, I would like to refer the reader to the survey of M. JONES *et al.* [2006].

Signed Distance Fields

In recent years, many methods have been presented to accelerate the exact evaluation of signed distance functions based on polygonal representations (see *e.g.*, SANCHEZ *et al.* [2012]). Even though the computational efficiency was drastically improved, the computation time still cannot fulfill the strict time constraints of applications such as interactive simulation or haptic rendering. Moreover, approximations to signed distance functions using precomputed SDFs can be efficiently queried and, therefore, serve as an excellent alternative. Due to the fact that discretizations of increasingly complex surfaces are very memory consuming, various methods focusing on a reduction of memory consumption were developed. One of the most popular methods is the *Adaptive Distance Field*s (ADFs) approach introduced by S. F. FRISKEN *et al.* [2000]. During the SDF's construction the underlying signed distance function is first sampled on a coarse grid and recursively refined using octree subdivisions as long as the deviation of trilinearly interpolated signed distance samples from the exact value exceeds a certain threshold. The method was later improved in terms of memory consumption and construction time by PERRY and S. F. FRISKEN [2001]. As a consequence of the refinement strategy, a large number of cells is required in regions where a trilinear discretization does not accurately represent the signed distance function. In order to further reduce the memory requirements, narrow band approaches were proposed by BÆRENTZEN [2002] and ERLEBEN and DOHLMANN [2008] that discretize only regions close to the object surface. A very cache efficient narrow band discretization approach for volumetric data on grid structures has been presented by MUSETH [2013]. This method is tailored to very large sparse data sets with grid resolutions of at least 8192^3 cells. In order to handle large data sets the approach uses a structure similar to a B+-tree to find the cells containing data. At this point, I would like to mention that a narrow band construction is directly applicable to the here proposed *hp*-adaptive SDFs. However, the present work focuses on a high quality representation of the SDF on the whole domain in order to quickly exclude possible contacts in the demonstrated application.

In contrast to using purely scalar valued SDFs several approaches were proposed that augment the discretization by additional geometric information. HUANG *et al.* [2001] propose a hybrid approach for distance field representation of polygonal meshes. Using a regular hexahedral grid, they store a list of triangles and the respective scalar distance value from the cell center to each triangle for each individual cell. This allows for an exact signed distance computation without any discretization errors as the polygonal representation is explicitly stored within the data structure. However, it is still necessary to perform expensive computations to evaluate distances to explicit polygons when querying the SDF. Moreover, explicitly storing the triangle lists results in a substantial memory overhead. N. MITCHELL *et al.* [2015] present multivalued signed distance fields where several cells might occupy a single volume portion of space. This allows to represent non-manifold features that cannot be represented by standard grid-based representations. As such, the approach is orthogonal to the aforementioned methods to represent more detail with less memory. Therefore, the method should also be compatible with the here presented grid based approach. In order to improve the representation of sharp features such as corners or hard edges while avoiding unnecessary refinement, several approaches were proposed. JU *et al.* [2002] store additional hermite data on the grid that represents exact intersection points and normals. Using an additional curvilinear offset grid, QU *et al.* [2004] align the mesh features with the second grid to improve the discretization. Similarly, BAERENTZEN [2005] uses an additional point cloud when reconstructing a mesh from an SDF in order to recover sharp features.

As opposed to discretizing the signed distance function using an axis-aligned hexahedral grid, J. WU and KOBBELT [2003] present a discretization approach based on binary space partitioning. In each subdivision step the splitting planes are aligned with geometric features in the input data in order to optimize the approximation. M. W. JONES [2004] completely avoids spatial subdivision but transforms the distance field with a vector distance transform using a specially defined predictor in order apply entropy compression to the distance data. By reducing the SDF data to a 2D height field projected onto a proxy geometry, OTADUY *et al.* [2004] as well as MOUSTAKAS *et al.* [2007] reduce the consumed memory. The main problem of both approaches is to define a suitable proxy geometry.

SDF-based Collision Handling

In the field of physics-based animation, SDFs allow for rapid distance queries between potentially colliding objects and are therefore especially well-suited for collision detection. Moreover, a contact normal for collision response can be directly deduced by computing the SDF's gradient as it points in the direction of the closest point on the object surface. Several works adopt the concept of SDF-based collision detection for rigid body simulations, *e.g.*, KAUFMAN *et al.* [2007], GLONDU *et al.* [2014], and XU *et al.* [2014]. BRIDSON *et al.* [2003], as well as FUHRMANN *et al.* [2003] use SDFs to resolve collisions between cloth and rigid objects. BARBIČ and JAMES [2008] present a method for haptic rendering involving data provided by SDFs. Image-based volume contacts proposed by FAURE *et al.* [2008] and ALLARD *et al.* [2010] are an alternative approach of capturing detailed contact geometry but require high-resolution sampling for precise contact handling. Therefore, B. WANG *et al.* [2012] apply, similar to the here presented method, an error estimation based on polynomials to guide the refinement of the spatial sampling. In order to improve efficiency and robustness of rigid body collision handling XU and BARBIČ [2014a] developed an SDF-based continuous collision detector. In this chapter, the applicability of novel hp-adaptive SDF representation in rigid body simulations with contacts is demonstrated. Please note, that the collision handling approach using the novel SDF representation is not limited to rigid body simulations and can also be applied for rigid-deformable and deformable-deformable collision detection when considering the modifications presented by MCADAMS *et al.* [2011].

Similar to several aforementioned methods, an adaptive construction algorithm for SDF generation is proposed here. In contrast to all previous approaches, polynomials are fit to the underlying signed distance function. The presented method is, to the best of my knowledge, the first approach that not only spatially subdivides (h-adaption) the grid but also improves the discretization by hierarchical augmentation of the polynomial basis with higher-order polynomials (p-adaption). As a consequence, the presented globally-adaptive, error-driven construction algorithm is able to generate memory efficient but highly accurate SDFs as shown in the results section.

hp-Adaptivity in Numerical Methods for PDEs

While the here presented approach is, to the best of my knowledge, the first approach on hp-adaptive generation of SDFs, hp-adaptivity is well-investigated in the field of numerical methods for PDEs, especially in the context of finite element solvers. In an early work, BABUŠKA *et al.* [1981] introduced the concept of p-adaptivity for a finite element solver for one- and two-dimensional PDEs and proved that its convergence rate is not worse (and in some cases even better) than h-adaptive approaches.

The concept was then combined with an h-adaptive approach and analyzed by BABUŠKA and SURI [1981] building the first hp-adaptive method in the field of FE analysis. For a discussion on further developments in the field until 1994, I would like to refer the reader to the survey of BABUŠKA and SURI [1994]. As hp-adaptive approaches refine the approximation in two distinct dimensions a suitable criterion is required to decide whether to refine in h- or p-direction. In this regard, W. F. MITCHELL and MCCLAIN [2014] compare several strategies to guide the hp-refinement. A very effective strategy to decide which refinement direction is likely to improve the approximation best was first proposed by SCHMIDT and SIEBERT [2000] for one-dimensional problems. They apply a refinement in p- and h-direction and choose the refinement direction that reduces the estimated remaining error the most. Unfortunately, for higher-dimensional problems this strategy is computationally very expensive as the numerical approximation has to be recomputed for each refinement step on the whole domain. In the application of SDF generation this problem is not present as the (discretized) field has only to be reconstructed in the currently considered cell which makes this strategy effective yet efficient. Similar to the basis polynomials in the here presented approach, HOUSTON *et al.* [2003] use Legendre polynomials as shape functions in an hp-adaptive finite element solver for hyperbolic conservation laws. They analyze the decay rate of the polynomial's Legendre expansion coefficients to estimate local regularity of the PDE's solution in each finite element and use this information to guide the refinement directions.

In contrast to methods for numerical solvers for PDEs, the here presented approach is targeted towards the construction of SDFs. A normalized, shifted Legendre polynomial basis is employed for discretization. The orthonormality property of the basis enables one to efficiently fit the basis functions to the exact input signed distance function without the requirement to solve a (usually required) linear equation system. Using the proposed polynomial basis, an efficient degree-based error estimator was developed and a novel refinement direction criterion that estimates the individual improvement of an h- or p-refinement step is proposed.

5.3 Signed Distance Field Construction

In this section, a novel hierarchical hp-adaptive SDF construction algorithm is presented. Given an initial grid on an axis-aligned bounding box representing a rectangular domain Ω the method aims to discretize a signed distance function Φ (*cf.* Equation (5.1)) implied by a corresponding surface descriptor, *e.g.,* a polygonal mesh. The algorithm can be divided into the following steps. In the first step, a coarse SDF is constructed by fitting low-order polynomials to Φ on each individual cell serving as an initial guess. In the second step, the error contributed by each cell is estimated by computing the quadratic distance between the approximating polynomial and its embedded lower order polynomial. Following a globally-adaptive top-down strategy, the cell that contributes the largest residual as candidate is selected for refinement in the third step and a novel decision criterion is applied in order to determine whether to apply h- or p-refinement. Finally, the third step is repeatedly performed until the aggregate residual of all cells tracked over the refinement process falls below the target error threshold. In the following sections a detailed explanation of each step will be given and the mechanisms and mathematical foundations to perform the discretization will be discussed.

Exact Signed Distance Computation

Given a triangular input mesh a method to determine the exact signed distance to the surface is required. Therefore, first the unsigned distance is computed by finding the closest triangle of the mesh and by subsequently evaluating the distance to the individual polygon. As a naïve search for the nearest triangle has linear complexity, the procedure is accelerated by construction of a bounding sphere hierarchy with a special traversal algorithm as proposed by SANCHEZ *et al.* [2012]. In order to determine the sign of the minimal distance, the approach of BÆRENTZEN and AANÆS [2005] is employed. The method uses the angle-weighted pseudo-normal test which only requires the evaluation of a single dot product with a precomputed surface normal.

Polynomial Fitting

In order to locally discretize the signed distance function, a multivariate polynomial of degree p is fitted to Φ on each individual cell. Given an arbitrary polynomial basis, this requires minimizing a quadratic distance measure between the polynomial approximation and the underlying signed distance function in order to find an optimal coefficient set for the basis polynomials. Mathematically, this results in the following quadratic minimization problem:

$$\begin{gathered}\min_{\mathbf{c}_e} R_e(\mathbf{c}_e)\\ R_e = \int\limits_{\Omega_e} \frac{1}{2}(f_e - \Phi)^2 d\boldsymbol{\xi}, \quad f_e = \mathbf{c}_e \cdot \mathbf{P}_e\\ \mathbf{P}_e = \{P_e^{\boldsymbol{\rho}}\},\ \mathbf{c}_e = \{c_e^{\boldsymbol{\rho}}\}\\ \boldsymbol{\rho} = (\rho_\xi, \rho_\eta, \rho_\zeta),\ 0 \leq \rho_\xi + \rho_\eta + \rho_\zeta \leq p,\end{gathered} \tag{5.2}$$

where R_e represents the half squared error to the exact signed distance function, Ω_e the domain of the eth cell, f_e the polynomial approximation of order p, respectively. Furthermore, $\mathbf{P}_e$ and $\mathbf{c}_e$ denote the polynomial basis vector and cell coefficient vector, respectively. $\boldsymbol{\rho}$ describes the polynomial degree in each direction of the corresponding basis polynomial. The solution to the quadratic minimization problem then corresponds to the solution of the linear equation system

$$\begin{gathered}\mathbf{A}_e\mathbf{c}_e = \mathbf{b}_e,\\ \mathbf{A}_e = \int\limits_{\Omega_e} \mathbf{P}_e \left(\mathbf{P}_e\right)^T d\boldsymbol{\xi}, \quad \mathbf{b}_e = \int\limits_{\Omega_e} \mathbf{P}_e \Phi d\boldsymbol{\xi}.\end{gathered} \tag{5.3}$$

which yields the desired coefficient set $\mathbf{c}_e$. The SDF can then be queried at point $\boldsymbol{\xi}$ by evaluating the fitted approximation $f_e(\boldsymbol{\xi})$.

Polynomial Basis and Hierarchical p-Refinement

Obviously, the underlying polynomial basis affects the structure and condition number of matrix $\mathbf{A}_e$. Moreover, the dense linear equation system grows when the degree of the polynomial basis is increased. For these reasons, a basis that is orthogonal on the corresponding cell is employed. The usage of this basis diagonalizes the matrix in Equation (5.3). The fact that Legendre polynomials have the property to be orthogonal on the interval $[-1, 1]$ makes them attractive for the construction of a higher-dimensional orthogonal basis. In order to generalize the Legendre basis to be orthogonal on an arbitrary interval, the coordinates have to be shifted accordingly. Consequently, a polynomial tensor-product basis based on shifted normalized Legendre polynomials is constructed that keeps the system well-conditioned and diagonalizes the generally dense linear system. The polynomial basis is then defined by

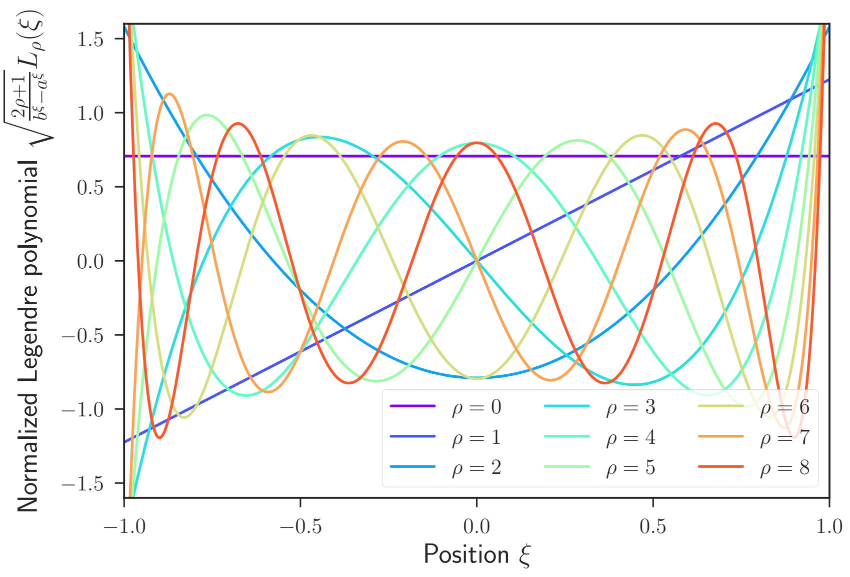


Figure 5.1: First nine (one-dimensional) shifted, normalized Legendre polynomials used for discretizing the signed distance function on cell interval $[a_{\xi}, b_{\xi}] = [-1, 1]$.

$$\begin{aligned} P_e^{\boldsymbol{\rho}}(\boldsymbol{\xi}) &= \prod_{x\in\{\xi,\eta,\zeta\}} \sqrt{\frac{2\rho_x+1}{b_e^x - a_e^x}} L_{\rho_x}(x') \\ L_p(x) &= \frac{1}{2^p}\sum_{l=0}^{p}\binom{p}{l}^2 (x-1)^{p-l}(x+1)^l \\ &= \frac{1}{p}\left((2p-1)\; x\; L_{p-1}(x) - (p-1)\; L_{p-2}(x)\right), \end{aligned} \tag{5.4}$$

where a_e^x and b_e^x are the minimum and maximum coordinate of the cell e in x-direction with shifted coordinate $x' = \frac{2}{b_e^x - a_e^x}x - \frac{b_e^x + a_e^x}{b_e^x - a_e^x}$. The one-dimensional portions of the Legendre basis are depicted in Figure 5.1. Due to the orthonormality of the basis, *i.e.*, $\int\limits_{\Omega_e} P_e^{\boldsymbol{\rho}} P_e^{\boldsymbol{\rho}^*}\,\delta\boldsymbol{\xi} = \delta_{\rho_\xi\rho_\xi^*}\delta_{\rho_\eta\rho_\eta^*}\delta_{\rho_\zeta\rho_\zeta^*}$, the solver matrix becomes the identity matrix, *i.e.*, $\mathbf{A}_e = \mathbf{I}$, where δ_{ij} denotes the Kronecker-δ. Consequently, the linear equation system (5.3) reduces to

$$\mathbf{c}_e = \int\limits_{\Omega_e} \mathbf{P}_e \Phi d\boldsymbol{\xi}. \tag{5.5}$$

Besides the fact that the requirement to compute and assemble $\mathbf{A}_e$ vanished, the diagonality implies that there is no coupling between the coefficients. This is especially advantageous as only new coefficients have to be computed when the approximation's polynomial degree is increased. As the approximation's polynomial degree can simply be augmented by computation of desired coefficients in $\mathbf{c}_e$, the basis can be considered to be *hierarchical*. By definition of Equation (5.2), the number of entries contained in vectors $\mathbf{P}_e$ and $\mathbf{c}_e$ is $n_c(p) = \frac{1}{6}(6+11p+6p^2+p^3)$. Please note that instead of a polynomial basis fulfilling $0 \leq \rho_\xi + \rho_\eta + \rho_\zeta \leq p$ the complete set of polynomials such that $0 \leq \max(\rho_\xi, \rho_\eta, \rho_\zeta) \leq p$ can be used for discretization. However, this strategy turned out to be less efficient as the number of resulting vector entries then grows faster ($n_c(p) = (p+1)^3$) resulting in less granular refinement steps.

In order to finally fit the basis polynomials to the underlying signed distance function, the integral equation describing the coefficient vector (*cf.* Equation (5.5)) has to be evaluated. Unfortunately, the only knowledge about the properties of the integrand one has, is that it is a continuous (but not necessarily smooth) function. Common approaches for the numerical integration of a-priori unknown functions include locally or globally adaptive, multi-dimensional numerical integration rules, *e.g.*, adaptive Gauss quadrature or Monte-Carlo integration using importance sampling. However, the convergence of these methods for the application to Equation (5.5) may suffer from two major issues. The first issue is the smoothness of the integrand. While the polynomials are smooth and therefore infinitely often differentiable, Φ is only guaranteed to be continuous for two-manifold geometries and usually contains discontinuities in its derivatives. Possible sources for these 'kinks' are sharp features in the underlying geometry or ambiguities in the signed distance function when the closest point on the surface is not unique. An example for the second case is the non-differentiability in the signed distance function at the center of a sphere where all surface points have the exact same distance to its center. This, finally, results in a large number of expensive Φ evaluations during numerical integration. In order to compute the integral sufficiently well and in an acceptable amount of time, it is heuristically approximated using multi-dimensional Gauss-Legendre quadrature of order $4p$, where p is the highest polynomial degree contained in $\mathbf{P}_e$. Using this heuristic, no artifacts or major issues were experienced. Moreover, the achieved results demonstrate that the method is able to generate very accurate SDFs using the described strategy.

Hierarchical h-Refinement

In contrast to increasing the polynomial order of the approximation, spatial subdivision can be an effective alternative for refining the SDF. Especially in regions where the underlying signed distance function is not smooth and therefore has low regularity, h-adaption is known to be more effective [W. F. MITCHELL and MCCLAIN 2014]. A simple but very reasonable explanation for this is that the very smooth polynomials are not suitable to represent 'kinks' in the function while more low-order polynomials are better at capturing these features. In order to realize the spatial subdivision, an octree is maintained for each of the base cells. After subdividing a cell into eight subcells corresponding to the next octree level, polynomials are again fitted to the exact signed distance function by means of solving Equation (5.5) and the coarse approximation is rejected. At this point I would like to stress the fact that an initial coarser approximation is not superfluous as it is essential for error estimation and the decision whether to apply h- or p-refinement in the further process.

Error Estimation

In order to steer the error-driven refinement process, the discretization error ϵ_e has to be computed over the domain of each individual cell e. Theoretically, it is possible to directly approximate the exact quadratic error $R_e(\mathbf{c}_e)$ using numerical integration because Φ can be evaluated exactly at any point on the domain. This, however, results in a significant computational effort for two reasons. Firstly, exact signed distance evaluations are expensive because they require to traverse the acceleration data structure and to compute geometric distances to multiple triangles. Therefore, the number of Φ evaluations should be kept to a minimum. Secondly, a numerical computation of $R_e(\mathbf{c}_e)$ with sufficient accuracy is hard as the integrand is in general locally non-smooth. Therefore, static numerical integration rules result in poor accuracy while adaptive techniques require an unacceptably large number of function evaluations. Please note that it was initially intended to approximate the exact error, but it was discovered that neither accuracy nor performance are acceptable in this case.

As a robust alternative, the cell-wise error can be estimated using the currently available approximation. More specifically, the estimation is based on the difference of the current degree p approximation compared to a lower order approximation of degree $p-1$:

$$\begin{aligned}\epsilon_e &= \int\limits_{\Omega_e} \left(\mathbf{c}_e \cdot \mathbf{P}_e - \mathbf{c}_e^* \cdot \mathbf{P}_e^*\right)^2 d\boldsymbol{\xi} \\ &= \int\limits_{\Omega_e} \left(\sum_{i+j+k=p} c_e^{(i,j,k)} P_e^{(i,j,k)} \right)^2 d\boldsymbol{\xi} \\ &= \sum_{i+j+k=p} \sum_{\alpha+\beta+\gamma=p} c_e^{(i,j,k)} c_e^{(\alpha,\beta,\gamma)} \int\limits_{\Omega_e} P_e^{(i,j,k)} P_e^{(\alpha,\beta,\gamma)} d\boldsymbol{\xi} \\ &= \sum_{i+j+k=p} |c_e^{(i,j,k)}|^2,\end{aligned} \tag{5.6}$$

where $\mathbf{c}_e^* = \{c_e^{\boldsymbol{\rho}^*}\}$ and $\mathbf{P}_e = \{P_e^{\boldsymbol{\rho}^*}\}$ are coefficient and polynomial vector of the embedded approximation of order $p-1$ of cell e with $0 \leq \rho_\xi^* + \rho_\eta^* + \rho_\zeta^* \leq p-1$. Due to the hierarchical basis the $p-1$ approximation is directly embedded in the current approximation. Moreover, the integral error measure can be evaluated analytically as all coefficients can be factored out while the remaining integral factors become either exactly one or zero as a consequence of the orthonormal basis. This finally results in a simple sum over the squared constant coefficients of the degree p basis polynomials. Please note that using smaller p approximations for a-posteriori error estimation is common practice in the finite element community and has proven to be an effective approach [W. F. MITCHELL and MCCLAIN 2014].

Construction Algorithm

In this section the novel construction algorithm based on the previously introduced error estimator and refinement strategies will be described. In general, the approach can be interpreted as an error-driven globally adaptive construction following a top-down strategy. The abstract procedure is outlined in Algorithm 5.1 and the reader will be guided through each step of the construction process.

Algorithm 5.1 hp-adaptive SDF construction.

```
Require: $\Phi, n_\xi, n_\eta, n_\zeta, \tau, \Omega, p_{\max}, l_{\max}$
 1: $\epsilon \leftarrow 0$
 2: $n \leftarrow n_\xi n_\eta n_\zeta$
 3: pending $\leftarrow$ priority_queue{}
 4: for $e \leftarrow 0$ to $n$ do
 5:     fit_polynomial($e$, $\Phi$, 2)                  // Fit polynomial of order 2 to
 6:                                                     // each base cell $e$ Eq. (5.5)
 7:     $\epsilon_e \leftarrow$ estimate_error($e$)      // Equation (5.6)
 8:     $\epsilon \leftarrow \epsilon + \epsilon_e$
 9:     pending.push({$e$, $\epsilon_e$})
10: while not pending.empty() and $\epsilon > \tau$ do
11:     {$e$, $\epsilon_e$} $\leftarrow$ pending.pop()
12:     $\{p, l\} \leftarrow$ {degree($e$), level($e$)}
13:     $\mu_e \leftarrow$ estimate_improvement_p($e$)   // Equation (5.8)
14:     $\nu_e \leftarrow$ estimate_improvement_h($e$)   // Equation (5.9)
15:     refine$_p \leftarrow p < p_{\max}$ and ( $1 == l_{\max}$ or $\mu_e > \nu_e$ )
16:     refine$_h \leftarrow l < l_{\max}$ and not refine$_p$
17:     if refine$_p$ then
18:         fit_polynomial($e$, $\Phi$, $p+1$)          // Equation (5.5)
19:         $\epsilon \leftarrow \epsilon - \epsilon_e$
20:         $\epsilon_e \leftarrow$ estimate_error($e$)  // Equation (5.6)
21:         $\epsilon \leftarrow \epsilon + \epsilon_e$
22:         pending.push({$e$, $\epsilon_e$})
23:     if refine$_h$ then
24:         children $\leftarrow$ subdivide($e$)         // Octree subdivision
25:         $\epsilon \leftarrow \epsilon - \epsilon_e$
26:         for $j \in$ children do
27:             fit_polynomial($j$,$p$)                  // Equation (5.5)
28:             $\epsilon_j \leftarrow$ estimate_error($j$)  // Equation (5.6)
29:             $\epsilon \leftarrow \epsilon + \epsilon_j$
30:             pending.push({$j$, $\epsilon_j$})
```

The algorithm expects the exact signed distance function Φ, an initial grid consisting of $n_\xi \times n_\eta \times n_\zeta$ cells on a rectangular domain Ω, the maximum refinement degree $p_{\max}$, the maximum octree depth $l_{\max}$, and the target error threshold τ as input. Further, the global error on the domain will be tracked using the total error ϵ in the course of the construction (*cf.* line 1).

The main idea of the construction algorithm is to maintain a priority queue which yields the index of the cell contributing the largest individual error in each iteration as a candidate for refinement. In line 1 to 3 the total error variable ϵ and the priority queue are initialized and the total number of base cells in the coarse initial grid is computed. In the initialization loop (lines 4 to 9) a polynomial of degree two is fitted to the underlying signed distance function on each individual cell, the contributed error is estimated, the error is accumulated in the total error variable, and the cell index based on its error contribution is inserted into the priority queue. The core part of the algorithm is the refinement loop described in lines 10 to 30. The loop refines the discretization until the error falls below the target error threshold τ and as long as refinable cells exist. After retrieving the element contributing the highest individual error from the priority queue, it has to be decided whether to spatially subdivide the cell or to increase its approximation's polynomial degree. If the cell has reached its maximum refinement level, only the degree may be increased and vice versa, such that no further criterion is required. Otherwise, the improvement that either p- or h-adaption yield is estimated. This is done by individually applying both refinement strategies and estimating the remaining error on the h-adaption induced subcells. Following this strategy, the following hp-decision criterion is developed:

$$\begin{cases} \text{adapt } p & \text{if } \mu_e > \nu_e \\ \text{adapt } h & \text{otherwise,} \end{cases} \tag{5.7}$$

$$\mu_e = \frac{1}{n_c(p+1) - n_c(p)} \left(\epsilon_e - \alpha\, \epsilon_e^{p+1}\right), \tag{5.8}$$

$$\nu_e = \frac{1}{7n_c(p)} \left(\epsilon_e - 8 \max_{c \in \mathcal{C}_e} \epsilon_c\right), \tag{5.9}$$

where $\mathcal{C}_e$ is the set of child cells resulting from the octree subdivision of e and $\alpha > 0$ an error scaling parameter. The error improvement per additional degree of freedom μ_e for p-refinement is computed based on the scaled error of the $(p+1)$-polynomial defined on the coarse cell. Analogously, a measure ν_e for the improvement corresponding to h-refinement is computed based on the scaled maximum error of the spatially subdivided order p polynomial measured on each of the subdomains on the finer octree level. The criterion decides in favor of p-adaption if the former improvement is greater than the latter. The reason for preferring the criterion over simply measuring which adaption would result in the greater improvement is the following. h-adaption is favored if the approximation on any of the potential subcells gains more accuracy from h-adaption compared to p-adaption. Moreover, as the used error measure is degree based, the algorithm tends to underestimate the remaining error for p-refinement. Therefore, the decision is additionally biased towards h-adaption using $\alpha = 8$ in order to counteract over-refinement in p-direction. Otherwise, the algorithm tends to drastically increase the polynomial degree in the first few steps as this improves the approximation on average over the coarse cell very well while there is potentially only a small improvement on some of the octree subdomains. This would force at least the same degree on the subcells resulting from subsequent h-adaptions. Consequently, many unnecessary degrees of freedom arise, leading to high memory consumption and computational effort for both construction and interpolation. At this point I would like to point out that the strategy to scale the p-error estimate for balancing the refinement is also common practice in the field of hp-adaptive finite element analysis [W. F. MITCHELL and MCCLAIN 2014]. Finally, lines 18-22 and 24-30 describe how the polynomial degree is increased and how the current cell is spatially subdivided, respectively. Then, the total error is updated and the resulting cells' indices together with the respective individual errors are inserted into the priority queue.

Please note that the total error is accumulated over the whole construction process. To avoid numerical errors due to the accumulation, the error estimate is stored on each individual cell and the total residual $\epsilon = \sum_e \epsilon_e$ is recomputed every 1000 iterations.

Nearness Weighting

For some applications of SDFs a comparably higher accuracy near the object's surface may be desired while regions far away from the surface are less interesting. Therefore, a weighting factor to compute a new transformed nearness weighted error estimate

$$\epsilon_e^* = \kappa_e \epsilon_e$$

is employed. This modification will artificially decrease the hp-refinement in regions far away from the object's surface. In order to find a suitable cell-wise coefficient κ_e, a measure encoding the distance to the object surface is required. Building on the fact that Φ provides the point-wise shortest distance to the surface, the average distance represented by a cell e is $(1/V_e)\int_{\Omega_e} \Phi d\boldsymbol{\xi}$ where V_e denotes the cell volume. As expensive Φ evaluations should be avoided, the current approximation f_e can be used instead. Then, a polynomial weighting factor

$$\kappa_e = \left(1 - \frac{1}{V_e d}\left|\int\limits_{\Omega_e} f_e d\boldsymbol{\xi}\right|\right)^{\theta}$$

can be modeled, where d and θ denote the construction domain's diagonal length and weighting exponent, respectively. Additionally dividing the distance measure by d normalizes it and ensures that $\kappa_e \in [0,1]$ as long as the domain fully contains the object surface, *i.e.*, $\mathcal{B} \subseteq \Omega$. However, due to potentially strong deviations of the approximation, κ_e may lie outside of the interval. In this case the factor can be simply clamped to $[0,1]$. If an even stronger decreasing weighting is desired, an exponential factor can be used instead:

$$\kappa_e = \exp\left(-\frac{\theta}{V_e d}\left|\int\limits_{\Omega_e} f_e d\boldsymbol{\xi}\right|\right).$$

Please note that if nearness weighting is used, the criterion described by Equations (5.7)-(5.9) must be modified accordingly.

5.4 Enforcing Weak Continuity

The previously presented algorithm describes a method to fit multivariate polynomials to the underlying signed distance function for each individual cell such that the global error is minimized efficiently. However, there is no guarantee that the resulting SDF is continuous over shared faces (interfaces) of neighboring cells. This is usually not a problem if the target error τ is chosen sufficiently small. If, however, a rather coarse approximation is desired, the 'jumps' in the discretization may pose a problem. Therefore, an approach using quadratic programming is presented in this section to enforce weak continuity over cell interfaces.

Generally, it is possible to project the discontinuous discretization onto a conforming discrete space. A method for the construction of a conforming hp-adaptive discretization was for example proposed by STOLFO *et al.* [2016]. The cell-wise decoupled approximation can be projected onto such a conforming discrete space, *e.g.*, using an L_2 projection. But since the polynomials are already optimally fit to the signed distance function with respect to the quadratic error measure R_e, one should aim to maintain the initial discretization as much as possible. Therefore, such a projection would potentially result in an uncontrollable loss of accuracy in order to directly enforce continuity. In contrast, better control of how much of the initial accuracy is lost for the sake of improving continuity can be gained by weak enforcement of continuous transitions.

Interface Error Measure

First an integral error measure is defined in order to quantify the discontinuity of the SDF between two neighboring cells. Let i be an interior face of the adaptive grid with i_l and i_r denoting the indices of its incident cells. Then, the integral error measure is defined as

$$\begin{aligned} E_i(\mathbf{c}_{i_l}, \mathbf{c}_{i_r}) &= \int\limits_{\Gamma_i} \frac{1}{2} \left(f_{i_l} - f_{i_r}\right)^2 dA \\ &= \frac{1}{2} \begin{pmatrix} \mathbf{c}_{i_l}^T & \mathbf{c}_{i_r}^T \end{pmatrix} \underbrace{\int\limits_{\Gamma_i} \begin{pmatrix} \mathbf{P}_{i_l}\mathbf{P}_{i_l}^T & -\mathbf{P}_{i_l}\mathbf{P}_{i_r}^T \\ -\mathbf{P}_{i_r}\mathbf{P}_{i_l}^T & \mathbf{P}_{i_r}\mathbf{P}_{i_r}^T \end{pmatrix} dA}_{\mathbf{M}_i} \begin{pmatrix} \mathbf{c}_{i_l} \\ \mathbf{c}_{i_r} \end{pmatrix}, \end{aligned} \tag{5.10}$$

where $\Gamma_i = \Omega_{i_l} \cap \Omega_{i_r}$ and where dA denotes the integration variable for surface integration. The matrix integral $\mathbf{M}_i$ in Equation (5.10) can be exactly evaluated using Gaussian quadrature of corresponding order. Moreover, due to the chosen polynomial basis, $\mathbf{M}_i$ can be evaluated analytically when the cells adjacent to an interface have the same cell size and therefore correspond to the same h-refinement depths. Each block contained in $\mathbf{M}_i$ consists of the outer product of two polynomial basis vectors. Let $\mathbf{P} = \{P_i\}$ and $\mathbf{P}^* = \{P_j^*\}$ be the normalized, shifted Legendre polynomial basis vectors of two neighboring cells, respectively. Since the polynomial basis vectors were constructed using a tensor-product, each component can be multiplicatively decomposed into factors only dependent on a single coordinate, *i.e.,* $P_i(\xi, \eta, \zeta) = P_{i,\xi}(\xi)P_{i,\eta}(\eta)P_{i,\zeta}(\zeta)$. Without loss of generality, let us assume that the normal of the common face points in ξ-direction. Let us further assume that both cells have the same size and are aligned with each other in η- and ζ-direction. Each block entry in $\mathbf{M}_i$ is constructed using an integral over the outer product of the basis vectors (*cf.* Equation (5.10)):

$$\int\limits_{\Gamma} \mathbf{P}\mathbf{P}^{*T} dA = \int\limits_{a^\eta}^{b^\eta} \int\limits_{a^\zeta}^{b^\zeta} \mathbf{P}\mathbf{P}^{*T} d\eta d\zeta.$$

Given the previously stated assumptions, the entry in the ith row and jth column of the matrix block can be analytically evaluated as follows:

$$\begin{aligned} \left[\int\limits_{\Gamma} \mathbf{P}\mathbf{P}^{*T} dA\right]_{ij} &= \int\limits_{a^\eta}^{b^\eta} \int\limits_{a^\zeta}^{b^\zeta} P_{i,\xi}P_{i,\eta}P_{i,\zeta}P_{j,\xi}^*P_{j,\eta}^*P_{j,\zeta}^* d\eta d\zeta \\ &= P_{i,\xi}P_{j,\xi}^* \underbrace{\int\limits_{a^\eta}^{b^\eta} P_{i,\eta}P_{j,\eta}^* d\eta}_{=\delta_{ij}} \underbrace{\int\limits_{a^\zeta}^{b^\zeta} P_{i,\zeta}P_{j,\zeta}^* d\zeta}_{=\delta_{ij}} \\ &= P_{i,\xi}P_{j,\xi}^* \delta_{ij}. \end{aligned}$$

Finally, the global discontinuity error is defined as sum of the cell-individual errors, *i.e.,* $E = \sum_{i \in \mathcal{I}} E_i$ where $\mathcal{I}$ denotes the set of all inner faces.

Regularization

It is obvious that a minimization of the error energy E in the global coefficient vector $\mathbf{c}$ minimizes the "jumps" in the discretization. However, the solution of the optimization problem is not unique as there is an infinite number of global coefficient vectors minimizing the function. Therefore, the problem needs to be regularized to guarantee uniqueness and to moreover find a meaningful solution.

While regularizing the problem, the initial optimized discretization should be maintained as much as possible. Therefore, a per-cell regularization energy

$$\begin{aligned}\Psi_e(\mathbf{c}_e, \mathbf{c}'_e) &= \int_{\Omega_e} \frac{1}{2}\left(f_e - f'_e\right)^2 d\boldsymbol{\xi} \\ &= \frac{1}{2}\left(\mathbf{c}_e - \mathbf{c}'_e\right)^T \underbrace{\int_{\Omega_e} \mathbf{P}_e \mathbf{P}_e^T d\boldsymbol{\xi}}_{\mathbf{I}} \left(\mathbf{c}_e - \mathbf{c}'_e\right) \\ &= \frac{1}{2}\left(\mathbf{c}_e - \mathbf{c}'_e\right)^T \left(\mathbf{c}_e - \mathbf{c}'_e\right)\end{aligned}$$

is defined, where f'_e, $\mathbf{c}'_e$ and $\mathbf{I}$ denote the initial (discontinuous) approximation, the according initial coefficient vector and the identity matrix, respectively. Once again, choosing an orthonormal polynomial basis pays off as the integral part of the regularization energy vanishes. The regularization energy over the whole domain is then defined by $\Psi = \sum_{e\in\mathcal{E}} \Psi_e = \frac{1}{2}(\mathbf{c} - \mathbf{c}') \cdot (\mathbf{c} - \mathbf{c}')$, where $\mathcal{E}$ denotes the set of cells contained in the grid.

Optimization

Using the previously defined interface error measure and regularization energy, a convex minimization problem in the coefficient vector $\mathbf{c}$ can be formulated in order to compute an SDF with improved continuity

$$\begin{gathered}\min_{\mathbf{c}} \left(E(\mathbf{c}) + \beta\Psi(\mathbf{c}, \mathbf{c}')\right) \\ \Leftrightarrow \\ \left(\mathbf{M} + \beta\mathbf{I}\right)\mathbf{c} = \beta\mathbf{c}', \end{gathered} \tag{5.11}$$

where $\beta > 0$ denotes a regularization parameter and $\mathbf{c}'$ the coefficient vector of the discontinuous approximation. Furthermore, $\mathbf{M} = \partial^2 E/\partial \mathbf{c}^2$ is the second derivative of the interface error term and can be assembled from the block matrices $\mathbf{M}_i$. An interpretation of β is simple as a minimization results in a more or less arbitrary (but continuous) SDF for $\beta \to 0$ whereas the solution converges towards the initial discretization induced by the initial coefficient vector $\mathbf{c}'$ for $\beta \to \infty$. Because the objective function is quadratic in the coefficient vector $\mathbf{c}$ and convex, the (unique) solution to the minimization problem is equivalent to the solution of the linear equation system given by Equation (5.11). The problem can finally be solved using an arbitrary (sparse) linear solver. As the number of coefficients, *i.e.,* degrees of freedom, is very large for detailed SDFs, the memory requirements for direct solvers using matrix factorization can easily exceed the available memory capacity. Therefore, a conjugate gradient descent solver in combination with an incomplete Cholesky factorization for preconditioning was employed for the presented results.

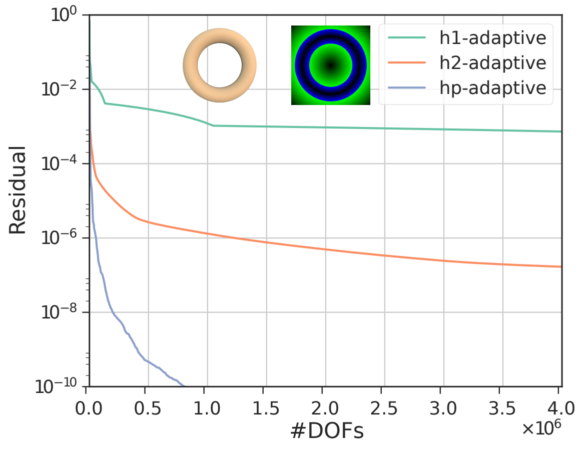


Figure 5.2: Comparison of the convergence behavior for a torus model of the novel *hp*-adaptive method with a pure octree-subdivision using linearly and quadratically fitted polynomials. #DOF encodes the number of polynomial coefficients required to enforce the corresponding residual.

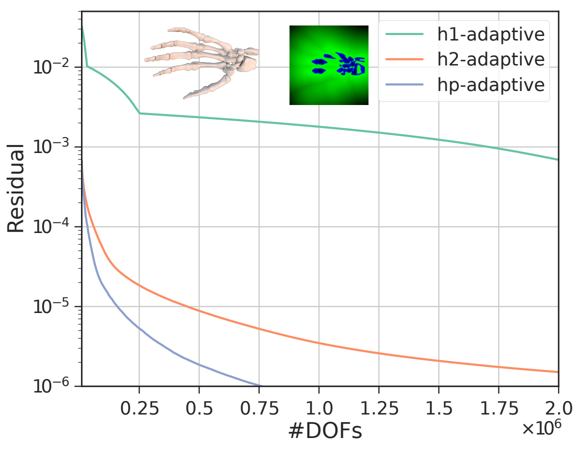


Figure 5.3: Convergence study of SDF construction for the skeleton hand model.

5.5 Results and Discussion

All computations in this section were carried out on two Intel Xeon E5-2697 processors with 2.7GHz, 30MB Cache, 12 cores per processor and 64GB RAM. The SDF construction was fully parallelized with Intel TBB and 48 threads were used in all computations. All deformable and rigid body simulations with contacts are based on the approaches proposed by BENDER *et al.* [2014b] and DEUL *et al.* [2014] implemented in the open-source library *PositionBasedDynamics* [BENDER 2017].

In summary, the presented results cover four types of experiments. Firstly, the convergence of the proposed method with respect to the number of coefficients was analyzed. Secondly, SDFs were generated for a variety of meshes and summarized in Table 5.2. Thirdly, various scenarios including rigid and deformable objects were simulated demonstrating the practical applicability of the novel approach for physically based simulation. Finally, the average time required to compute distance values with the novel SDF representation was measured. In the following paragraphs, each of these experiments will be described in detail.

Convergence and Refinement Analysis

Figures 5.2 and 5.3 illustrate the convergence graph during SDF construction for a torus and a skeleton hand model. The number of required coefficients ($\#DOF$) is shown on the abscissa while the estimated error (residual) from Equation (5.6) is displayed on the ordinate in logarithmic scale. The novel hp-adaptive approach was compared to pure octree-subdivision with linear ($h1$-adaptive) and quadratic ($h2$-adaptive) polynomials. Both examples show the superiority of the presented approach on the basis of the given error measure as it requires a fraction of the number of coefficients compared to the other methods. The curves' 'kinks', most visible in the curve of the $h1$-adaption, appear when all cells of a certain octree level are subdivided such that the decrease in the residual becomes suddenly smaller. I would like to stress the fact that the polynomials were, in all cases, constructed using the fitting approach (*cf.* Equation (5.2)) which yields the optimal solution in terms of the measured error. Using the traditional approach of sampling distance values within each cell would yield even worse results for the $h1$- and $h2$-adaptions. For further investigation, the leaf cells and their polynomial degree were visualized for an example slice as depicted in Figures 5.4 and 5.5. It can be noticed that h-refinement with low-order polynomials was primarily used in regions where Φ is non-differentiable while smooth regions are mainly represented by large cells with high polynomial degree.leftleft This exactly correlates with the desired behavior and demonstrates the meaningfulness and applicability of the hp-decision criterion (*cf.* Equation 5.7).

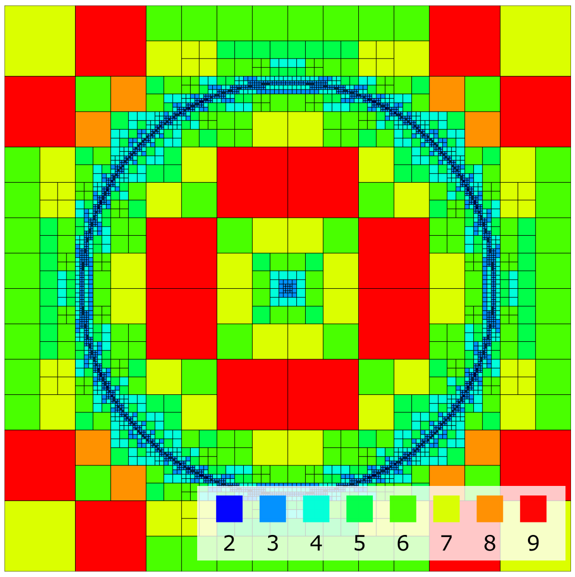


Figure 5.4: Torus degree plot. Visualization of octree cells with corresponding polynomial degrees according to the legend depicted in the figure.

In order to demonstrate the effect of nearness weighting, an SDF for the skeleton hand was constructed with exponential nearness weighting with $\theta = 30$. The according degree plot is illustrated in Figure 5.6. The result clearly shows how regions close to the object surface are strongly refined while regions far away correspond to a coarse discretization in comparison to the unweighted result in Figure 5.5. Moreover, the reduced field required approximately 72% less memory compared to the unweighted result.

Construction Statistics

Table 5.2 summarizes statistics on the input triangle meshes and the according SDF construction results. All input meshes were scaled to the unit box $[-1, 1]^3$ and the construction domain was enlarged by 10% while the refinement limits were chosen as $p_{\max} = 30$ and $l_{\max} = 10$. Additionally, polynomial nearness weighting with $\theta = 4$ was used for all examples. The mesh column shows the name of the mesh and its number of vertices and faces. The SDF column further contains the resolution of the initial construction grid, the time required for construction, the number of the resulting octree leaf cells, the distributions of degree and octree depth, the target error and the final memory consumption as well as a visualization of an exemplary slice of the SDF. The SDFs were stored in a data structure which essentially consists of four arrays. The first two arrays contain the polynomial coefficients in double-precision and a prefix-sum stating at which index in the coefficient array the coefficients of each cell start and how many coefficients belong to the respective cell. The remaining two arrays hold a child node index list containing the indices of the corresponding octree nodes stored in the last array.

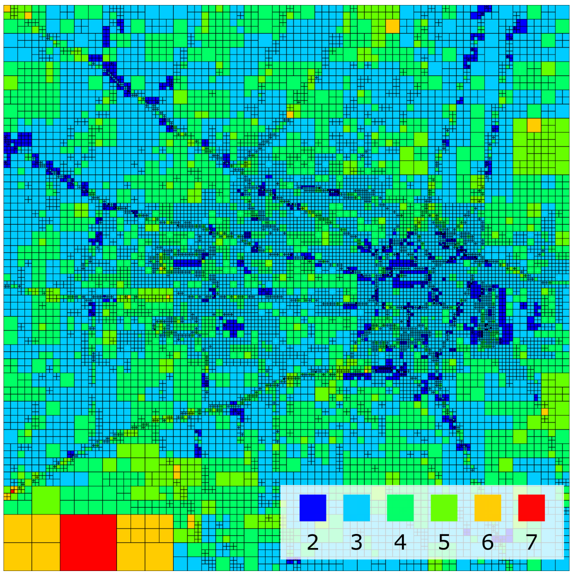


Figure 5.5: Skeleton hand degree plot. Visualization of octree cells with corresponding polynomial degrees according to the legend depicted in the figure.

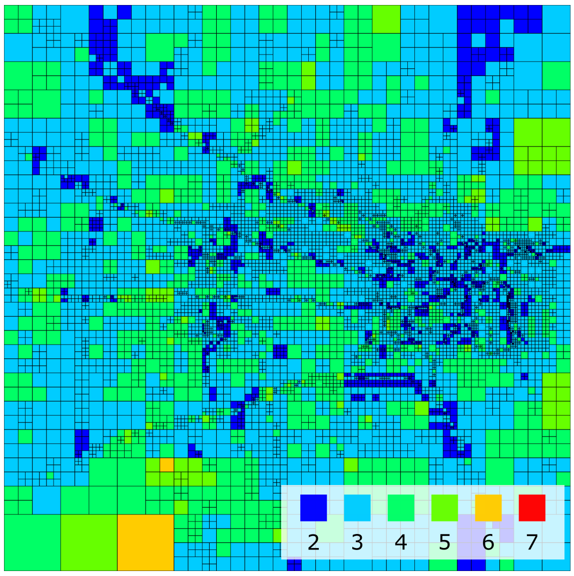


Figure 5.6: Skeleton hand degree plot of nearness-weighted result. Visualization of octree cells with corresponding polynomial degrees according to the legend depicted in the figure.

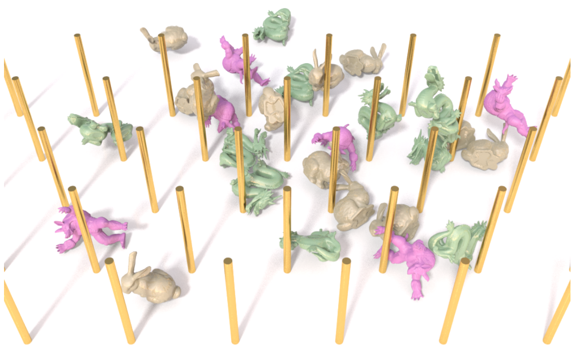


Figure 5.7: Complex rigid and deformable bodies slide down an inclined plane with obstacles.

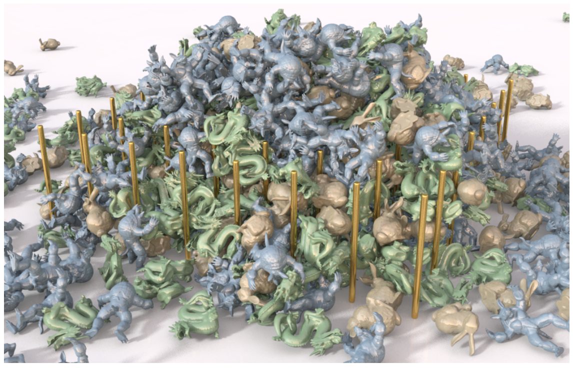


Figure 5.8: 800 rigid bodies fall onto a set of 64 poles having several thousand contacts per simulation step.

Collision Detection

The novel SDF representation was used in several physics-based simulations as collision detector in order to demonstrate the practical relevance of the approach. To realize the collision detector, the involved objects' surfaces were additionally point-sampled and organized in a *Bounding Sphere Hierarchy* (BSH). The BSH was constructed and traversed similar to the approach described by SANCHEZ *et al.* [2012]. As the SDF construction process can be interpreted as preprocessing step, all generated SDFs were serialized. Finally, the relevant SDFs were loaded prior to each simulation scenario and tested against the point samples of the other objects. Two experiments, depicted in Figures 5.7 and 5.8, were conducted. In both scenarios, several dynamic bodies collide with a static grid of poles. Each

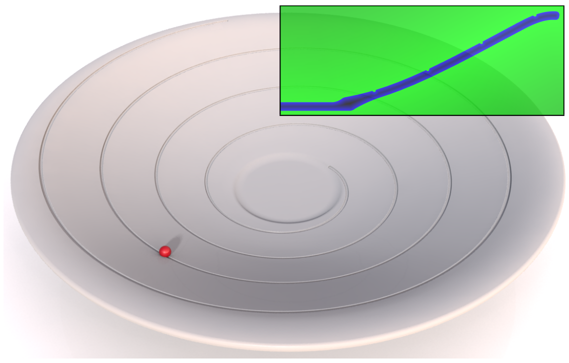


Figure 5.9: Dynamic simulation of a marble following a highly-detailed, helix shaped groove in a bowl.

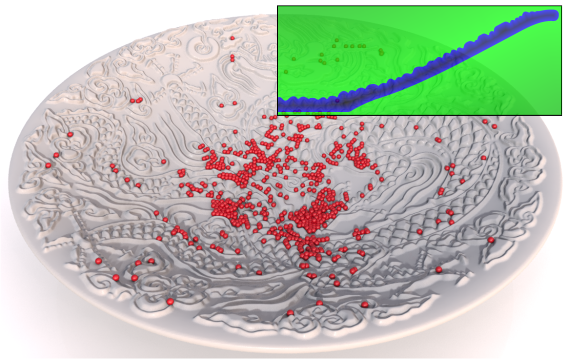


Figure 5.10: Collisions of 1000 marbles dropped into a bowl with highly complex structures are accurately resolved using the novel SDF representation.

body was sampled with approximately 10k sample points to perform the distance queries. In order to reduce the number of distance queries a BSH was implemented. The collision tests were further accelerated by parallelization of the collision test for each object pair. In the first scenario (*cf.* Figure 5.7) deformable and rigid bodies slide on an inclined plane. While SDFs for the rigid dragons and bunnies was used, collisions of the deformable armadillos were exclusively tested using the distance fields of the rigid bodies and obstacles. In the second experiment 800 rigid armadillo, bunny and dragon models were dropped onto a set of 64 poles. After simulating a time interval of 25 seconds the average and maximum number of contacts per step were 8007 and 15050, respectively. More than 11600 contacts per time step were observed in the finally resting state. The collision detection including signed distance field queries and BSH traversal required a computation time of 158 ms on average per time step. Here, it should be mentioned that an accurate detection based on geometric vertex-triangle and edge-edge tests would not have been feasible in a comparable computation time on the CPU as the scenario manages models with a total number of more than 132M triangles. In further scenarios, two finely structured bowls as illustrated in Figure 5.9 and 5.10 were simulated. In the first bowl a marble rolls on a helical groove whereas 1000 comparably small marbles were dropped into the second bowl. In both scenarios the contact information between the marbles and the bowls is very accurate while the highly-detailed surfaces are flawlessly represented by the SDF. Figure 5.11 shows a marble rolling on a marble run. Please note that the geometry is very thin and small compared to its bounding box. The presented method was still able to construct a very accurate SDF that required only a small amount of memory (*cf.* Table 5.2). Finally, a sheet of cloth covering the Stanford dragon was simulated as depicted in Figure 5.12. The features of the dragon surface are still clearly visible when they are silhouetted against the sheet.

Continuity Optimization

The post-processing optimization step for enforcing weak continuity was performed on an SDF of the Stanford dragon ($\tau = 5 \times 10^{-7}$, exponential nearness weighting $\theta = 20$). As depicted in Figure 5.13 (left) the SDF was first generated with a moderate target error. While the discretization captures the macroscopic shape of the underlying geometry well, especially regions with high-frequent features

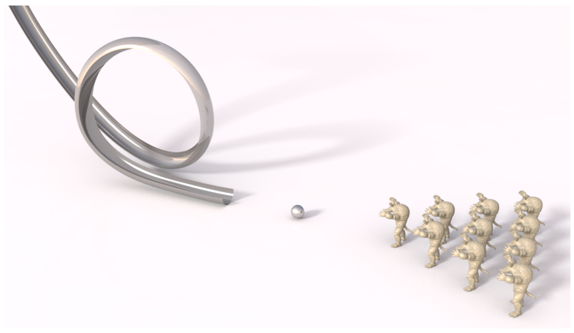


Figure 5.11: Dynamic simulation of a marble run with subsequent armadillo bowling.

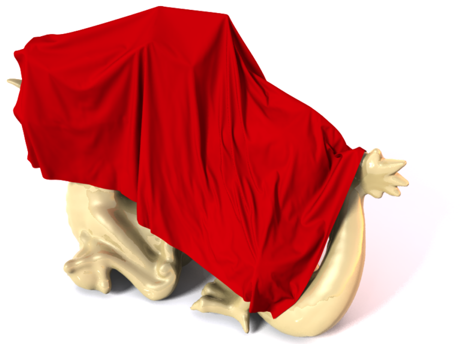


Figure 5.12: A sheet of cloth represented by 320k triangles is dropped on the Stanford dragon. The mesh's characteristic features are outlined due to the accurate SDF representation serving as collision detector.

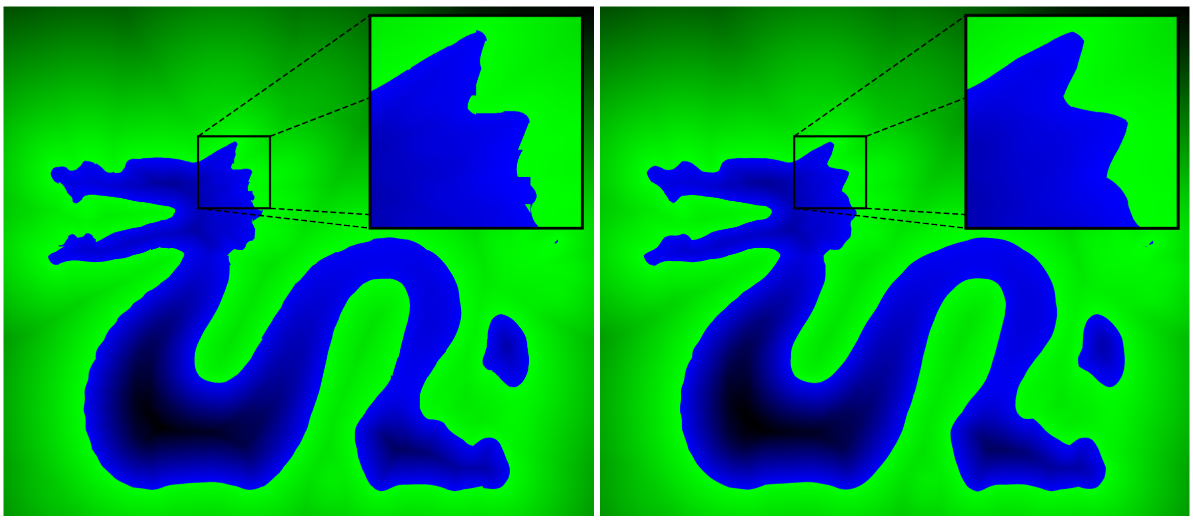


Figure 5.13: Discretization of the Stanford dragon (approx. 5.4MB) with target error $\tau = 5 \times 10^{-7}$ and exponential nearness weighting ($\theta = 20$). Left: SDF with subtle artifacts due to discontinuities on element interfaces. Right: SDF after continuity optimization with regularization parameter $\beta = 3$.

suffer from the generally discontinuous piecewise approximation. In order to improve the continuity of the field, the SDF was optimized as explained in Section 5.4 with regularization parameter $\beta = 3$. The SDF consisted of $1.3M$ coefficients and the sparse solver matrix contained $178M$ non-zero entries. The construction of the matrix took $47s$ and the incomplete Cholesky decomposition took $58s$. Both procedures were not parallelized. The actual solve of the equation system with a conjugate gradient solver took $44s$ with 88 iterations using a parallelized implementation of the matrix vector product. The result illustrated in Fig. 5.13 (right) clearly shows that the approximation continuity is greatly improved while the general shape and features are maintained.

Improving the continuity of the field can also be beneficial when the SDF is applied in collision handling. The marble run scenario described in the previous paragraph was simulated using an unoptimized but very accurately discretized SDF (*cf.* Table 5.2). The scenario was resimulated using a less

h-level	Query time $[ns]$
0	169
1	202
2	226
3	245
4	276
5	322
6	380
7	611
8	1157

(a)

p-level	Query time $[ns]$
1	169
2	194
3	238
4	276
5	336
6	397
7	483
8	577
9	697

(b)

Table 5.1: Performance measurements. A single cell was uniformly refined using either p- or h-adaption. The resulting discrete SDF was then queried using random samples and the average time for a single query was computed.

accurate but optimized SDF ($\tau = 10^{-8}$, #cells $= 22.3k$, 4.36MB, $\beta = 8$, exponential nearness weighting with $\theta = 20$). Using this more compact but post-processed representation the scenario was reproduced with the marble rolling smoothly until it reaches the end of the track and finally resulting in a dynamic simulation of comparable quality.

Distance Query Performance

In order to measure the performance of queries into the novel discrete SDF, a field was randomly sampled with several thousand points and the resulting measured values were averaged. For the armadillo and structured bowl SDFs, this resulted in approximately 4.76×10^{-4} ms and 7.16×10^{-4} ms, respectively. If the SDF-gradient was additionally requested, the queries took 7.34×10^{-4} ms and 7.84×10^{-4} ms on average. In order to analyze the individual effect of h- or p-refinement on the query performance the required query time for an SDF initially consisting of a single cell with linear polynomials that was zero to eight times h-refined or up to nine times p-refined was analyzed. The results are depicted in Tables 5.1a and 5.1b. Thanks to the recursive form of the Legendre polynomials (*cf.* Equation (5.4)) their evaluation can be accelerated by reusing redundant terms from order 0 to order $p - 1$ during the distance evaluation as well as the gradient computation.

5.6 Conclusion

In this chapter, a novel method to hierarchically construct higher-order SDFs was presented. The approach efficiently fits shifted, orthonormalized Legendre polynomials to the exact signed distance function in a hierarchical manner. Spatial adaptivity was implemented using octree subdivision. A new hp-decision criterion using degree-based error estimation was developed in order to steer the adaption in the refinement process. By comparing the method to traditional purely spatially adaptive approaches, it was demonstrated that the criterion-controlled refinement algorithm can greatly improve convergence. Furthermore, a nearness weighting approach was introduced that modifies the estimated error in order to focus the refinement on regions close to the surface of the underlying object. Moreover, an optimization-based post-processing step was proposed, that weakly enforces continuity over element interfaces. It was demonstrated that the method is able to produce very accurate SDFs for complex geometries while consuming only a small amount of memory. Moreover, the SDFs were shown to be very well-suited for the detection of contacts and collisions in physically based simulations as they implicitly contain information about the penetration depth and contact normal.

The proposed technique also has limitations. In its current version the method only features isotropic refinement. However, an anisotropic adaption strategy could significantly reduce the memory requirements while still maintaining an accurate approximation. This could potentially be realized by using a k-d-tree for spatial refinement and axis-dependent degree adaption for p-refinement. A fundamental problem is that for two-dimensional objects embedded in three dimensions, such as cloth or shells, inside and outside cannot be distinguished. For that reason the method is not applicable for handling cloth-cloth contacts or self-intersections of cloth. Collisions between touching surfaces – a scenario which occurs in cutting or fracture simulations – cannot easily be handled by simply discretizing the signed distance. For that reason, it should be investigated if the proposed method can be extended to a non-manifold representation similar to the work of N. MITCHELL *et al.* [2015].

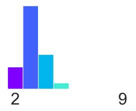
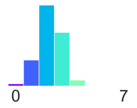
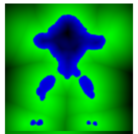
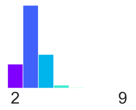
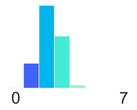
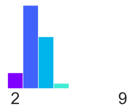
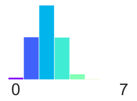
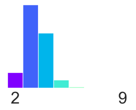
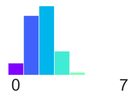
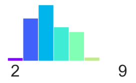
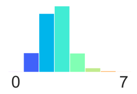
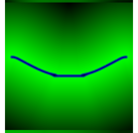
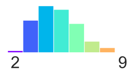
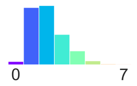
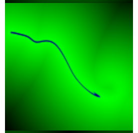
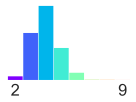
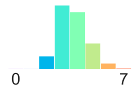
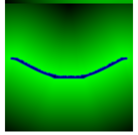
Mesh			Signed Distance Field							
Name	#Vert.	#Faces	Base Grid	Constr. Time	#Cells	Degree	Depth	ϵ, τ	Memory	Field
Armadillo	173k	346k	6^3	270s	71k			10^{-6}	10.6 MB	
Bunny	34.8k	69.6k	6^3	301s	55k			10^{-6}	8.4 MB	
Dragon	40k	80k	10^3	1245s	299k			10^{-7}	46.9 MB	
Hand	66.2k	132.5k	10^3	833s	159k			10^{-7}	25 MB	
Helix Bowl	164.9k	329.8k	4^3	423s	159k			5×10^{-9}	34.9 MB	
Marble Run	27k	53k	4^3	312s	87k			5×10^{-9}	18.4 MB	
Structured Bowl	2.05M	4.1M	4^3	4121s	1.25M			5×10^{-9}	250 MB	

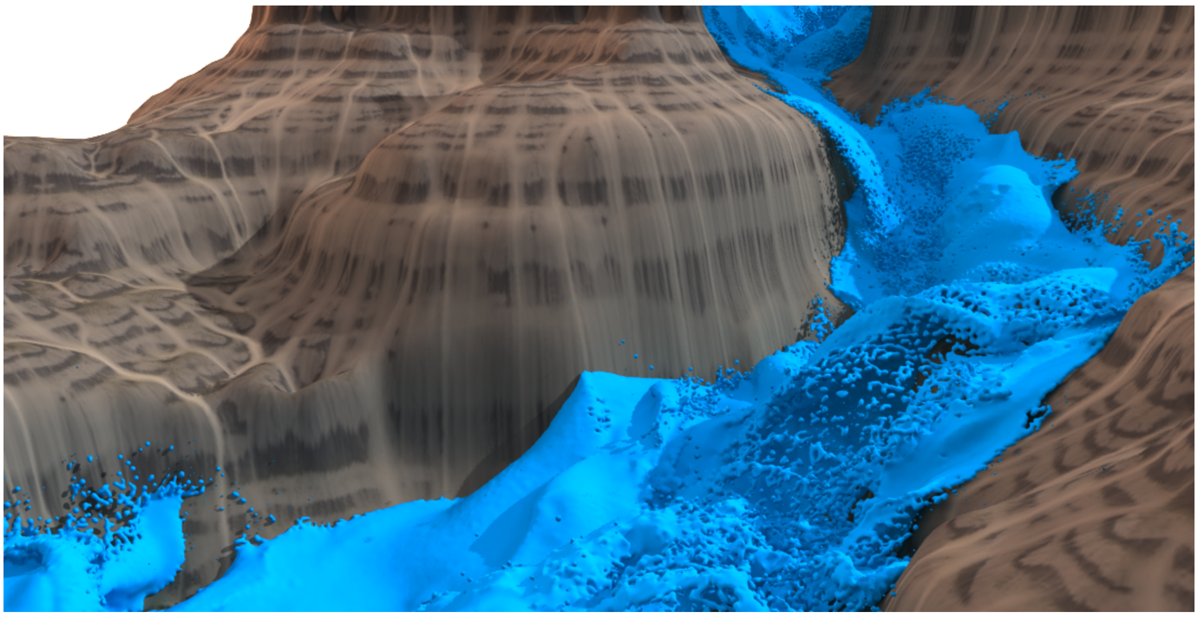
Table 5.2: Construction statistics. The mesh columns show object names and the corresponding numbers of vertices and faces. The SDF columns contain initial grid resolution, required time for construction, number of octree leaf cells, normalized histograms capturing the volume-fraction of the domain occupied by cells of the corresponding degree or octree depth, the (enforced) target error and the final memory consumption as well as a slice image of the SDF.

Part *III*

SPH Fluid Simulation with Implicit Boundary Handling

Chapter 6

Simulation of Incompressible Fluids using Smoothed Particle Hydrodynamics



In this chapter a novel method for the simulation of incompressible fluids based on the semi-discretization technique SPH is presented. While the concept of SPH was originally introduced to solve astrophysical problems in open space [GINGOLD and MONAGHAN 1977] it has become an increasingly popular method for the simulation of fluids in the fields of engineering [M. LIU and G. LIU 2010] and computer animation [IHMSEN *et al.* 2014b]. In this context it is usually used to spatially discretize the *Navier-Stokes Equations* (NSE) in Lagrangian coordinates. The basic idea is to use a set of particles in combination with so-called smoothing kernel functions to discretize a continuum and all differential operators contained in the according PDE. Most approaches for the spatial discretization of the NSE such as the FEM, *Finite Volume Method* (FVM) or *Finite Difference*s (FDs) usually discretize the space into polyhedral meshes in Eulerian coordinates such that the mesh stays constant over time. However, describing the NSE in Eulerian coordinates has several disadvantages as *e.g.,* enforcing conservation of mass becomes less trivial and as the convection term includes a non-linear differential operator. Mesh based numerical solvers for the discretization of the NSE in Lagrangian coordinates are, however, extremely complicated because a fluid constantly undergoes heavy deformations and topological changes. Under these circumstances the solver relies on complex remeshing operations to maintain a meaningful discretization. In contrast, the SPH approach is inherently mesh-free in the sense that the continuum is represented by a set of particles in combination with so-called smoothing kernel functions. With the help of these kernels the occuring differential operators can be directly discretized and (almost) no attention has to be paid to heavy deformations and topological changes within the fluid. Over the years the SPH concept has been widely adopted in the field of computer graphics and has become an important technique for the efficient and robust simulation of complex visual and physical fluid effects. The research in recent years has pushed the computational efficiency of the methods such that we are able to simulate scenarios with highly complex fluid effects featuring millions of particles.

Most fluids we encounter in our daily life are almost incompressible. The incompressibility therefore implies an additional constraint in the continuum formulation as already introduced in Chapter 2. This constraint can be formulated either on position or velocity level such that both formulations are mathematically equivalent. However, numerical solvers suffer from a numerical drift in the position if only the velocity-level constraint is satisfied. Similarly, the solver can become unstable if only the position-level constraint is satisfied. Despite these issues, the vast majority of SPH based methods for the simulation of incompressible fluids satisfies either position- or velocity-level constraint accepting the implied consequences. However, the continuity equation for incompressible flow demands to maintain a constant density and a divergence-free velocity field. This chapter proposes a combination of two novel implicit pressure solvers enforcing both a low volume compression as well as a divergence-free velocity field. While a compression-free fluid is essential for realistic physical behavior, a divergence-free velocity field drastically reduces the number of required solver iterations and increases the stability of the simulation significantly. Thanks to the improved stability, the method can handle larger time steps than previous approaches. This results in a substantial performance gain since the computationally expensive neighborhood search has to be performed less frequently. The efficiency, robustness, and scalability of the method is demonstrated in a variety of complex simulations including scenarios with millions of particles.

6.1 Introduction

In the area of computer graphics SPH is an important meshless Lagrangian approach to simulate complex fluid effects. The SPH formalism allows one to efficiently compute a certain quantity associated with a fluid particle by considering only a finite set of neighboring particles. One of the most challenging research topics in the field of SPH methods is the simulation of incompressible fluids. The fact that most fluids we encounter in nature feature incompressible behavior, proves that enforcing incompressibility is essential to produce realistic animations for a wide range of materials. Therefore, the following sections introduce a novel *Divergence-free Smoothed Particle Hydrodynamics* (DFSPH) method to simulate incompressible fluids.

In order to model incompressible fluids the model used in this work is based on the incompressible, isothermal Navier-Stokes equations in Lagrangian coordinates. Let us therefore recall this PDE introduced in Chapter 2, Equation (2.16)

$$\frac{D\rho}{Dt} = 0 \quad \Leftrightarrow \quad \nabla \cdot \mathbf{v} = 0 \tag{6.1}$$

$$\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \mathbf{v} + \frac{\mathbf{f}}{\rho}, \tag{6.2}$$

where $\rho, p, \nu, \mathbf{v}$ and $\mathbf{b}$ denote density, pressure, kinematic viscosity, velocity and body force density field, respectively. According to Equation (6.1) an incompressible fluid satisfies the *divergence-free condition* and therefore has a divergence-free velocity field. Based on the observation that the density must not change over time, *i.e.*, $\dot{\boldsymbol{\rho}} = 0$, and based on the continuity equation (2.8) the equivalence stated in Equation (6.1) easily follows. Therefore, in theory, a fluid with a divergence-free velocity field is incompressible, as it maintains constant density. However, in practice, incompressibility cannot be guaranteed in SPH simulations by only enforcing the divergence-free condition. The numerical time integration causes numerical errors which accumulate over time. The resulting density deviations are not considered in the divergence-free condition and therefore a creeping volume compression cannot be avoided. In order to avoid the numerical error to accumulate over time the second condition $\rho = \text{const}$ implying

$$\rho - \rho_0 = 0$$

is employed. This condition will be in referred to as *constant density condition*. To summarize, one can say that a position-level solver for the incompressible, isothermal Navier-Stokes equations must ensure a divergence-free velocity field *and* should provide an additional stabilization to counteract numerical errors. This stabilization can be realized by enforcing the constant density condition.

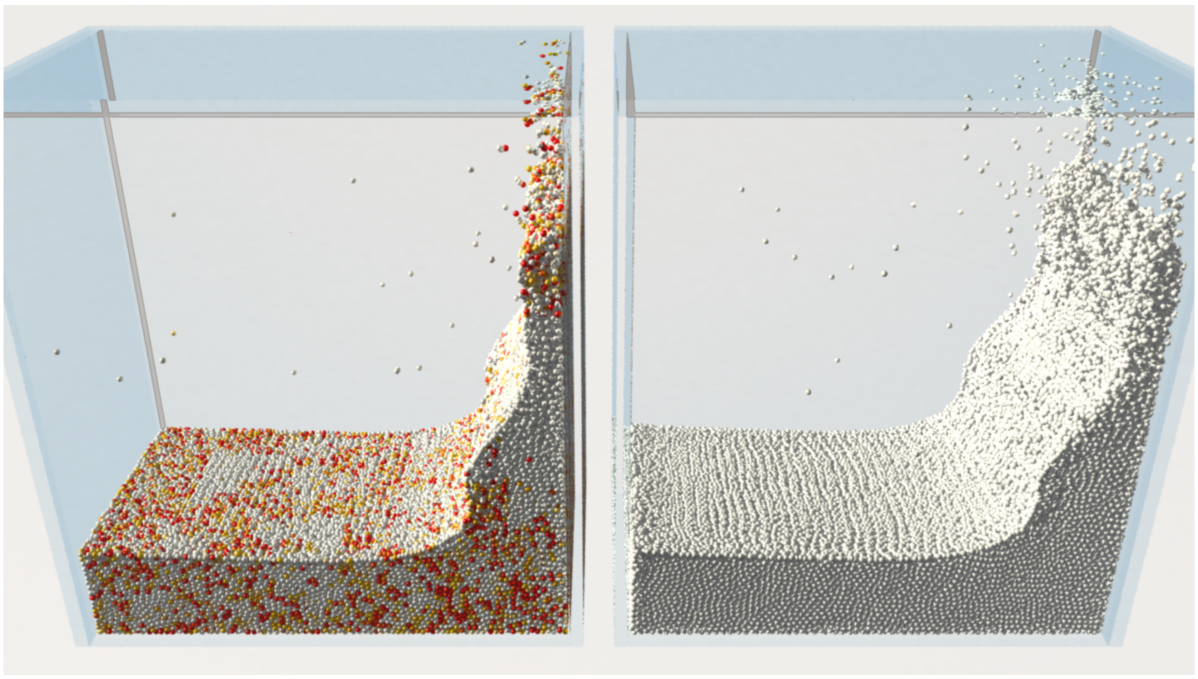


Figure 6.1: Velocity divergence comparison of IISPH (left) and DFSPH (right) in a breaking dam simulation with 125k particles. The divergence errors are color coded, where white represents the minimum and red is the maximum error value.

In recent years several implicit SPH solvers which compute pressure forces to counteract volume compression were proposed. To date, the most efficient SPH pressure solvers only consider the constant density condition. Since this condition solely depends on particle positions, the divergence-free condition is generally not fulfilled (*cf.* Figure 6.1, left) as demanded by the continuity equation for incompressible flow. A correction of velocity divergence is considered only by a few SPH approaches so far. However, they are either comparatively slow or cannot maintain a small density error.

In this chapter a novel stable and efficient SPH method for incompressible flow is introduced. In contrast to the vast majority of previous methods, it satisfies both the constant density and the divergence-free condition. The method combines two pressure solver steps. The first step enforces a divergence-free velocity field (*cf.* Figure 6.1, right) while the constant density condition is satisfied in a second solver step. This combination has several advantages in SPH simulations. Firstly, the stability of the simulation increases significantly, especially in simulations with fast moving particles. The improved stability allows the solver to perform larger time steps than previous approaches. This yields a considerable gain in computational performance as the neighborhood search has to be performed less frequently. Secondly, the number of iterations required by the constant density solver decreases significantly when a divergence-free velocity field is enforced. Finally, the divergence-free field allows a more accurate computation of the maximum time step size as the commonly used *Courant-Friedrichs-Levy* (CFL) condition depends on the maximum particle velocity.

In the last years different efficient constant density solvers have been proposed, *e.g., Predictive-Corrective Incompressible SPH* (PCISPH) [SOLENTHALER and PAJAROLA 2009] or *Implicit Incompressible SPH* (IISPH) [IHMSEN *et al.* 2014a]. Nevertheless, here a new approach is proposed that operates analogously to the novel divergence-free solver, and therefore benefits from reusing precomputed coefficients. The fluid simulation based on the combined solver clearly outperforms state-of-the-art SPH methods and can be more than 20 times faster in typical scenarios.

6.2 Related Work

In this section, approaches related to the presented method will be briefly discussed. For a more detailed discussion, I would like to refer to reader to the work of IHMSEN *et al.* [2014b] for SPH-based simulation approaches.

Equation of State Based Solvers

In the pioneering work of Monaghan the first approaches to simulate fluids using SPH discretizations were proposed [MONAGHAN 1989; MONAGHAN 1992; MONAGHAN 1994]. Using an *Equation of State* (EOS), deviations from a given rest density were penalized using a stiffness coefficient in order to weakly enforce incompressibility. Several years later, an EOS-based approach for fluid simulation was introduced to the computer graphics community by M. MÜLLER *et al.* [2003] and later extended by B. ADAMS *et al.* [2007] for spatially adaptive SPH discretizations. In order to restrict the maximum compression using an EOS-based solver, BECKER and TESCHNER [2007] precomputed a scenario-dependent stiffness coefficient. However, this may lead to stiff differential equations which restrict the time step size for explicit integration schemes drastically.

Iterative Incompressibility Solvers Based on Splitting

An alternative strategy, often referred to as the concept of splitting, is to first discretize the Navier-Stokes equations in time and to solve the spatial differential terms sequentially. Two of these steps are to compute the fluid pressure after advecting particles only based on non-pressure forces. Initially, the concept was applied to EOS-based solvers [Chorin 1968; Bridson 2008] only but soon combined with implicit pressure solvers as discussed in the following.

Solenthaler and Pajarola [2009] developed an iterative pressure solver based on the splitting concept to approximately limit the maximum density error to a user-defined tolerance. Later, the method was extended by rigid-fluid coupling [Akinci *et al.* 2012] and a novel surface tension model [Akinci *et al.* 2013a]. Macklin and M. Müller [2013] proposed *Position Based Fluids* (PBF) – a similar approach that iteratively corrects particle positions to enforce incompressibility. A velocity-level formulation using holonomic constraints based on rigid body mechanics was proposed by Bodin *et al.* [2012]. However, due to numerical errors, their regularization approach and velocity-level correction, they report density errors of up to 17 %. In contrast to the discussed iterative approaches the here presented method does not only enforce incompressibility on velocity or density level but considers both. Another method eliminating the compression on both levels was recently proposed by Kang and Sagong [2014]. They extended the work of Macklin and M. Müller [2013] by additionally projecting the velocity field onto a the divergence-free state. However, the velocity field is not guaranteed to contain zero divergence after a time step as particle positions are updated subsequently to the velocity projection. Moreover, their extension leads to a considerable overhead in comparison to the underlying method of Macklin and Müller while the here presented method yields large speedup factors of up to one order of magnitude, especially in case of large time steps (*cf.* Table 6.2).

Pressure Projection

Another popular choice to enforce incompressibility is to project the velocity field onto a divergence-free state by solving the PPE. This method is particularly popular for mesh based discretizations of the NSE in Eulerian coordinates [Bridson 2008]. A PPE derived from an approximate pressure projection was proposed by Cummins and Rudman [1999]. They solved the resulting linear system using either a conjugate gradient method or a multigrid approach treating the particles as finest level grid. Foster and Fedkiw [2001] developed a semi-Lagrangian method where incompressibility is enforced on the simulation grid. They counteracted mass dissipation using a level set and freely moving inertialess particles. Further hybrid approaches using particles and a background grid were developed in the following years [Zhu and Bridson 2005; Raveendran *et al.* 2011; Ando *et al.* 2013]. An interesting and more elaborate approach was proposed by Losasso *et al.* [2008]. They solve the PPE on a background grid not only to maintain zero divergence but to enforce a predefined target density. Sin *et al.* [2009] constructed a Voronoi discretization of the PPE from point-samples in each step for solving. In contrast to the discussed approaches, the here presented method does not rely on any background grid. For that reason it does not require to maintain memory consuming data-structures which is especially advantageous for large scenarios.

In contrast to solve the PPE on a background grid, it may be directly solved on a meshless discretization. Hu and N. Adams [2007] proposed an incompressible SPH-based approach for multiphase flows with multistep time integration. In a first halfstep the density error is eliminated using a position-altering iterative gradient descent solver. In the second halfstep, errors in velocity divergence are similarly corrected. However, they applied their method exclusively to non-complex two-dimensional scenarios of up to 14,400 particles. A local Poisson solver extending the work of

SOLENTHALER and PAJAROLA [2009] was proposed by HE *et al.* [2012a]. The solver enforces both a divergence-free velocity and constant density field. As the integration domain for the local solves is not necessarily equal or a subset of the local kernel function support, they have to recompute the particle neighbors. Moreover, a recomputation of particle neighborhoods is required in each solver iteration resulting in a considerable computational effort. Compared to the method of Solenthaler and Pajarola they are able to achieve a small speedup factor of approximately 1.5 while the here presented approach achieves speedup factors of more than 20. IHMSEN *et al.* [2014a] proposed a method based on a PPE that iteratively eliminates the density error to 0.1% or even less. However, they do not consider resolving velocity divergence which leads to a large number of solver iterations compared to the here presented method and can even cause artifacts under certain circumstances (*cf.* Section 6.8). Recently, an incompressible particle-based approach using power-diagrams was proposed by DE GOES *et al.* [2015]. In each simulation step they construct a Voronoi diagram induced by the particle positions in order to enforce incompressibility. However, the here presented approach clearly outperforms the method as the diagram construction is very time consuming.

6.3 SPH Discretization

The novel fluid simulator is based on the incompressible, isothermal Navier-Stokes equations introduced in Section 6.1. However, in this work the viscous term on the right hand side of Equation (6.2) is omitted. Instead the XSPH variant of SCHECHTER and BRIDSON [2012] is employed in order to simulate low viscous fluids like water.

Equation (6.2) is discretized using the SPH concept which is introduced in the following. Equation (6.1) represents the velocity-level incompressibility conditions. The presented solver satisfies these conditions by enforcing the divergence-free condition and the constant density condition as described in Sections 6.5 and 6.6, respectively.

Following the standard SPH approach scalar and vector fields are approximated and rewritten in discrete form as described in the following. Given a function $f : \Omega \to \mathbb{R}^n$ that maps a position vector $\mathbf{x}$ in the problem domain $\Omega \subset \mathbb{R}^3$ to a scalar or vectorial value, it can be rewritten using the Dirac delta identity and approximated by replacing the delta distribution with a kernel function $W : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ with compact support:

$$\begin{aligned} A(\mathbf{x}) &= \iiint\limits_{\Omega} A(\mathbf{x}^*)\, \delta(\mathbf{x}-\mathbf{x}^*) d\mathbf{x}^* \\ &\approx \iiint\limits_{\mathcal{D}_\mathbf{x}} A(\mathbf{x}^*)\, W(\|\mathbf{x}-\mathbf{x}^*\|, h) d\mathbf{x}^* \\ &= \iiint\limits_{\mathcal{D}_\mathbf{x}} \frac{A(\mathbf{x}^*)}{\rho(\mathbf{x}^*)}\, W(\|\mathbf{x}-\mathbf{x}^*\|, h)\, \rho(\mathbf{x}^*)\, d\mathbf{x}^*, \end{aligned} \tag{6.3}$$

where h represents the smoothing length of the kernel and $\mathcal{D}_\mathbf{x}$ the spherical domain around $\mathbf{x}$ where $W(\|\mathbf{x}-\mathbf{x}^*\|) > 0$. In the following, the radius of the domain $\mathcal{D}_\mathbf{x}$ will be referred to as support radius r. More specifically, the kernel is chosen such that $\lim_{h\to 0} W(\|\mathbf{x}-\mathbf{x}^*\|, h) = \delta(\|\mathbf{x}-\mathbf{x}^*\|)$ subject to normalization $\iiint_\Omega W(\|\mathbf{x}-\mathbf{x}^*\|, h) d\mathbf{x}^* = 1$. The integral resulting from Equation (6.3) is then discretized into a sum over particle positions where each particle j carries a portion of the fluid mass m_j:

$$A(\mathbf{x}) \approx \sum_j \frac{m_j}{\rho_j} A(\mathbf{x}_j) W(\|\mathbf{x} - \mathbf{x}_j\|, h).$$

Please note that due to the fact that the kernel function W has compact support, only a small subset of the fluid particles located close to the evaluation point $\mathbf{x}$ contribute to the sum. Using efficient algorithms to determine the neighboring particles the number of evaluations to compute the sum can be minimized. The spatial derivative of $A(\mathbf{x})$ is determined by

$$\nabla A(\mathbf{x}) \approx \sum_j \frac{m_j}{\rho_j} A(\mathbf{x}_j) \left.\frac{\partial W(q,h)}{\partial q}\right|_{q=\|\mathbf{x}-\mathbf{x}_j\|} \frac{\mathbf{x} - \mathbf{x}_j}{\|\mathbf{x} - \mathbf{x}_j\|}. \tag{6.4}$$

When the presented discretization is applied to the density field the formula reduces to

$$\rho(\mathbf{x}) \approx \sum_j m_j W(\|\mathbf{x} - \mathbf{x}_j\|, h).$$

The pressure field of a fluid is often computed using the following EOS:

$$p_i = \frac{k\rho_0}{\gamma}\left(\left(\frac{\rho_i}{\rho_0}\right)^{\gamma} - 1\right),$$

where ρ_0 is the rest density and k and γ are stiffness parameters. In this work the parameter γ was set to 1 which is a common choice in computer graphics applications [DESBRUN and GASCUEL 1996; M. MÜLLER *et al.* 2003; IHMSEN *et al.* 2014b]:

$$p_i = k(\rho_i - \rho_0). \tag{6.5}$$

The pressure field can be directly used to determine pressure forces which counteract density deviations [MONAGHAN 1992]. However, a high stiffness coefficient k is required to keep the fluid sufficiently incompressible leading to systems of stiff differential equations. Therefore, implicit pressure solvers such as PCISPH [SOLENTHALER and PAJAROLA 2009] or IISPH [IHMSEN *et al.* 2014a] for the simulation of incompressible fluids were developed. Such solvers typically rely on solving a linear equation system to determine the pressure field. The implicit solve then allows a stable simulation with larger time steps resulting in a significant performance gain.

In the following sections a novel implicit SPH approach to simulate incompressible fluids will be presented. In contrast to previous methods it employs two solver steps: the *divergence-free solver step* (*cf.* Section 6.5) and the *constant density solver step* (*cf.* Section 6.6). The first solve enforces a divergence-free velocity field. Since density deviations resulting from numerical errors cannot be corrected in this way, as discussed earlier, the second solver step eliminates these density errors by satisfying the constant density condition. All in all, incompressibility and a divergence-free velocity field are enforced as demanded by Equation (6.1). In both solver steps the respective conditions are fulfilled for each neighborhood independently by computing a stiffness coefficient k_i (*cf.* Equation (6.5)) and the corresponding pressure forces. To obtain a global solution the solver steps process the neighboring particles in a parallel Jacobi fashion which is a common approach [SOLENTHALER and PAJAROLA 2009; MACKLIN and M. MÜLLER 2013; IHMSEN *et al.* 2014a].

Please note that in the following subscripts will be used to simplify notation. The subscripts denote that a function is evaluated at particle positions according to the subscript index in appearing order, *e.g.*, $\rho_i = \rho(\mathbf{x}_i)$ or $W_{ij} = W(\|\mathbf{x}_i - \mathbf{x}_j\|, h)$. If the subscript notation is moreover used in combination with a differential operator, it denotes the derivative of the function with respect to the particle position associated with the first index, *e.g.*, $\nabla\rho_i = \nabla\rho|_{\mathbf{x}=\mathbf{x}_i}$ or $\nabla W_{ij} = \frac{\partial W(q,h)}{\partial q}\Big|_{q=\|\mathbf{x}_i-\mathbf{x}_j\|} \frac{\mathbf{x}_i-\mathbf{x}_j}{\|\mathbf{x}_i-\mathbf{x}_j\|}$.

6.4 Simulation Step

Algorithm 6.1 DFSPH simulation step.

```
1:  function PERFORMSIMULATIONSTEP
2:      for all particles $i$ do                                   // init neighborhoods
3:          find neighborhoods $N_i(0)$
4:      for all particles $i$ do                                   // init $\rho_i$ and $\alpha_i$
5:          compute densities $\rho_i(0)$
6:          compute factors $\alpha_i(0)$
7:      while ($t < t_{\max}$) do                                   // simulation loop
8:          for all particles $i$ do
9:              compute non-pressure forces $\mathbf{F}_i^{adv}(t)$
10:         adapt time step size $\Delta t$ according to CFL condition
11:         for all particles $i$ do                               // predict velocities $\mathbf{v}_i^*$
12:             $\mathbf{v}_i^* = \mathbf{v}_i + \Delta t \mathbf{F}_i^{\text{adv}}/m_i$
13:         correctDensityError($\alpha$, $\mathbf{v}^*$)                       // $\rho^* - \rho_0 = 0$
14:         for all particles $i$ do                               // update positions
15:             $\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i^*$
16:         for all particles $i$ do                               // update neighbors
17:             find neighborhoods $N_i(t+\Delta t)$
18:         for all particles $i$ do                               // update $\rho_i$ and $\alpha_i$
19:             compute densities $\rho_i(t+\Delta t)$
20:             compute factors $\alpha_i(t+\Delta t)$
21:         correctDivergenceError($\alpha$, $\mathbf{v}^*$)                    // $\frac{D\rho}{Dt} = 0$
22:         for all particles $i$ do                               // update velocities
23:             $\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i^*$
```

This section is intended to abstractly outline the simulation step. A detailed description of the substeps is given in the following sections. The Navier-Stokes equations for incompressible fluids demand that the constant density condition and the divergence-free condition are satisfied at the end of a simulation step. In order to fulfill the first condition pressure forces are determined. These have to be integrated twice to get the required position changes. Afterwards the pressure forces are computed and integrated once to make the resulting velocity field divergence-free. Please note that performing the density stabilization before computing a divergence-free velocity field does not introduce any errors as both steps are repeatedly executed in a loop.

The simulation based on the novel method is outlined in Algorithm 6.1. At the beginning of each loop iteration all particle neighborhoods N_i are determined using compact hashing [IHMSEN *et al.* 2011]. Furthermore, the particle densities ρ_i and correction factors α_i (*cf.* Section 6.5) are computed. The factors α_i can be used in both solver steps and only depend on the current particle positions. For that reason they remain constant throughout the iterative process of the individual solver steps and it is sufficient to compute them only once in each simulation step. This reduces the computational effort associated with the solver iterations considerably.

In the first step of the simulation loop all non-pressure forces $\mathbf{F}^{\text{adv}}$, *i.e.*, gravitational, surface tension and viscosity forces, are computed. Then, the time step size is adapted according to the CFL condition [MONAGHAN 1992]

$$\Delta t \leq 0.4 \frac{d}{\|\mathbf{v}_{\max}\|},$$

where $\mathbf{v}_{\max}$ is the maximum particle velocity and where d is the particle diameter. Note that in contrast to PCISPH [SOLENTHALER and PAJAROLA 2009] no special treatment is required to perform adaptive time-stepping. In line 12 of the algorithm a predicted velocity $\mathbf{v}_i^*$ is determined for each particle by solely considering the non-pressure forces. This prediction and the precomputed factors α_i are then used by the constant density solver step to compute the pressure forces in order to correct the density error $\rho_i^* - \rho_0$ (*cf.* Section 6.6) for each neighborhood. The time integration in line 15 determines the new positions of the particles. Hence, all neighborhoods N_i, densities ρ_i and factors α_i have to be updated. Finally, the condition $\frac{D\rho_i}{Dt} = 0$ must be satisfied. This is done in line 21 where the divergence-free solver step determines pressure forces to make the velocity field divergence-free (*cf.* Section 6.5). In the last step the particle velocities are updated.

Note that the neighborhood information N_i, the densities ρ_i and the factors α_i must be computed only once per simulation step. These values are initialized before the simulation loop in lines 2-6 and updated once per time step in lines 16-20. The values are not determined in the beginning of the simulation loop as in previous methods because the solver steps are executed at different points in time.

6.5 Divergence-Free Solver Step

The goal of the divergence-free solver step is to satisfy the condition $\frac{D\rho}{Dt} = 0$ which means that the density must not change over time. From Equation (6.1) it follows that in this case the divergence of the velocity field $\nabla \cdot \mathbf{v}$ must be zero.

In order to enforce the condition $\frac{D\rho_i}{Dt} = 0$ for a particle i the solver determines a set of pressure forces which correct the divergence error in the neighborhood of the particle. The pressure force density of particle i is defined by

$$\mathbf{f}_i^p = -\nabla p_i. \tag{6.6}$$

The pressure gradient ∇p_i is determined by computing the spatial derivative of Equation (6.5) using the SPH approach (*cf.* Equation (6.4)), *i.e.*,

$$\nabla p_i = k_i^v \nabla \rho_i = k_i^v \sum_j m_j \nabla W_{ij},$$

where k_i^v is the stiffness parameter that has to be determined. In order to compute the symmetric pressure forces for each neighborhood, the force densities $\mathbf{f}_{j\leftarrow i}^p$ which act from the particle i on its neighboring particles j have to be considered. A set of symmetric pressure forces must satisfy the condition $\mathbf{f}_i^p + \sum_j \mathbf{f}_{j\leftarrow i}^p = \mathbf{0}$ which means that the forces sum up to zero which is required to conserve momentum. The pressure force densities $\mathbf{f}_{j\leftarrow i}^p$ for the neighboring particles are determined analogously to

Equation (6.6) except that the pressure must be differentiated with respect to position $\mathbf{x}_j$:

$$\mathbf{f}^p_{j\leftarrow i} = -\frac{\partial p_i}{\partial \mathbf{x}_j} = k^v_i m_j \nabla W_{ij}. \tag{6.7}$$

The current divergence error in particle i is determined using the SPH formulation of the divergence IHMSEN *et al.* [2014b]:

$$\frac{D\rho_i}{Dt} = \sum_j m_j(\mathbf{v}_i - \mathbf{v}_j)\nabla W_{ij}. \tag{6.8}$$

The goal of the divergence-free solver step is to compute a set of symmetric pressure forces to enforce $\frac{D\rho_i}{Dt} = 0$. These pressure forces cause the following velocity changes $\Delta\mathbf{v}_i = \Delta t\mathbf{f}^p_i/\rho_i$ and $\Delta\mathbf{v}_j = \Delta t\mathbf{f}^p_{j\leftarrow i}/\rho_i$. Because the velocity changes that eliminate the divergence are desired, the following equation results:

$$-\frac{D\rho_i}{Dt} = \Delta t \sum_j m_j \left(\frac{\mathbf{f}^p_i}{\rho_i} - \frac{\mathbf{f}^p_{j\leftarrow i}}{\rho_i}\right)\nabla W_{ij}. \tag{6.9}$$

Moreover, an equation for the stiffness parameter k^v_i can be deduced by using Equations (6.6) and (6.7) in Equation (6.9), *i.e.*,

$$\begin{aligned}
\frac{D\rho_i}{Dt} &= -\Delta t \sum_j m_j \left(\frac{\mathbf{f}^p_i}{\rho_i} - \frac{\mathbf{f}^p_{j\leftarrow i}}{\rho_i}\right)\nabla W_{ij} \\
&= \frac{\Delta t}{\rho_i}\sum_j m_j \left(k^v_i \sum_j m_j \nabla W_{ij} + k^v_i m_j \nabla W_{ij}\right)\nabla W_{ij} \\
&= k^v_i \frac{\Delta t}{\rho_i}\left(\left|\sum_j m_j \nabla W_{ij}\right|^2 + \sum_j |m_j \nabla W_{ij}|^2\right).
\end{aligned}$$

Solving for k^v_i yields:

$$k^v_i = \frac{1}{\Delta t}\frac{D\rho_i}{Dt} \cdot \underbrace{\frac{\rho_i}{\left|\sum_j m_j \nabla W_{ij}\right|^2 + \sum_j |m_j \nabla W_{ij}|^2}}_{\alpha_i}, \tag{6.10}$$

where α_i is a factor that solely depends on the current particle positions. If the pressure forces are computed using the resulting stiffness parameter k^v_i, the condition $\frac{D\rho_i}{Dt} = 0$ will be exactly fulfilled. This means that a divergence-free velocity field in the neighborhood of particle i is obtained. In order to enforce a globally divergence-free velocity field, the pressure forces for all particles are computed iteratively using parallel Jacobi iterations.

If particle i has a small number of neighbors, the denominator of α_i can be close to zero. Therefore, the denominator is clamped if it gets smaller than 10^{-6} to keep the simulation robust. Alternatively, the problem can be solved by adding a small constant to the denominator [MACKLIN and M. MÜLLER 2013] or using a reference configuration with a prototype neighborhood [SOLENTHALER and PAJAROLA 2009].

Algorithm 6.2 Divergence-free solver step

```
1: function CORRECTDIVERGENCEERROR($\alpha$, $\mathbf{v}^*$)
2:     while $\left(\left(\frac{D\rho}{Dt}\right)_{\text{avg}} > \eta^{\text{div}}\right) \vee (\text{iter} < 1)$ do
3:         for all particles $i$ do                                    // compute $\frac{D\rho}{Dt}$
4:             $\frac{D\rho_i}{Dt} = -\rho_i \nabla \cdot \mathbf{v}_i^*$
5:         for all particles $i$ do                                    // adapt velocities
6:             $k_i^v = \frac{1}{\Delta t}\frac{D\rho_i}{Dt}\alpha_i, \quad k_j^v = \frac{1}{\Delta t}\frac{D\rho_j}{Dt}\alpha_j$
7:             $\mathbf{v}_i^* := \mathbf{v}_i^* - \Delta t \sum_j m_j \left(\frac{k_i^v}{\rho_i} + \frac{k_j^v}{\rho_j}\right)\nabla W_{ji}$
```

The total force $\mathbf{f}_{i,\text{total}}^p$ for a particle i is the sum of all pressure forces:

$$\begin{aligned}
\mathbf{f}_{i,\text{total}}^p &= \frac{m_i}{\rho_i}\mathbf{f}_i^p + \sum_j \frac{m_j}{\rho_j}\mathbf{f}_{i\leftarrow j}^p \\
&= -\frac{m_i}{\rho_i}k_i^v \sum_j m_j \nabla W_{ij} + \sum_j \frac{m_j}{\rho_j} k_j^v m_i \nabla W_{ji} \\
&= -m_i \sum_j m_j \left(\frac{k_i^v}{\rho_i} + \frac{k_j^v}{\rho_j}\right)\nabla W_{ij}.
\end{aligned}$$

Note that the resulting symmetric pressure force is equivalent to the one introduced by MONAGHAN [1992].

In each iteration the stiffness parameters k_i^v depending on the factors α_i have to be determined. Fortunately, the factors can be precomputed before the iterative process as they solely depend on the current positions which yields computationally cheap iteration steps. Note that Equation (6.10) must be adapted for static boundary particles since in this case $\mathbf{f}_{j\leftarrow i}^p = 0$.

The divergence-free solver step is outlined in Algorithm 6.2. It terminates when the average density change rate falls below the threshold η^{div}. To improve the convergence of solver step, a "warm start" operating as explained in the following is used. The stiffness values k_i^v are summed up for each particle during the iterative process. Before the divergence-free solver step in the subsequent time step starts, line 7 is evaluated once for each particle using the resulting values.

6.6 Constant Density Solver Step

The constant density solver step minimizes the deviation $\rho - \rho_0$ of the current density to the rest density. Previous pressure solvers like PCISPH [SOLENTHALER and PAJAROLA 2009] or IISPH [IHMSEN *et al.* 2014a] analogously minimize the density deviation and can be combined with the here presented divergence-free solver step. However, in this work a new constant density solver was developed which works analogously to the divergence-free solver step. This has the advantage that the precomputed factors α_i can be reused which reduces the computational effort significantly. The solver uses a predictor-corrector scheme similar to PCISPH in order to determine particle positions for the next time step to correct the density error. However, in contrast to PCISPH, no prototype configuration with a filled neighborhood is precomputed to solve the system as this reduces the convergence rate.

Algorithm 6.3 Constant density solver

```
1: function CORRECTDENSITYERROR($\alpha$, $\mathbf{v}^*$)
2:     while $(\rho_{\text{avg}} - \rho_0 > \eta^\rho) \vee (\text{iter} < 2)$ do
3:         for all particles $i$ do                                // predict density
4:             compute $\rho_i^*$
5:         for all particles $i$ do                                // adapt velocities
6:             $k_i = \frac{\rho_i^* - \rho_0}{\Delta t^2}\alpha_i$,   $k_j = \frac{\rho_j^* - \rho_0}{\Delta t^2}\alpha_j$
7:             $\mathbf{v}_i^* := \mathbf{v}_i^* - \Delta t \sum_j m_j \left(\frac{k_i}{\rho_i} + \frac{k_j}{\rho_j}\right) \nabla W_{ij}$
```

The prediction of the density error $\rho_i^* - \rho_0$ is determined by integrating Equation (6.8) using an explicit Euler scheme

$$\rho_i^* = \rho_i + \Delta t \frac{D\rho_i}{Dt} = \rho_i + \Delta t \sum_j m_j (\mathbf{v}_i^* - \mathbf{v}_j^*) \nabla W_{ij}.$$

Analogous to Equation (6.9), the forces to correct this error are computed by solving

$$\rho_i^* - \rho_0 = \Delta t^2 \sum_j m_j \left(\frac{\mathbf{f}_i^p}{\rho_i} - \frac{\mathbf{f}_{j\leftarrow i}^p}{\rho_i} \right) \nabla W_{ij}.$$

The resulting stiffness parameter is determined by

$$k_i = \frac{1}{\Delta t^2} (\rho_i^* - \rho_0) \alpha_i.$$

Algorithm 6.3 outlines the implicit constant density solver step. Note that a warm start analogous to the one of the divergence-free solver step is performed to reduce computation time.

6.7 Efficient Kernel Computation

SPH methods typically use kernel functions which approximate the Gaussian kernel. Sometimes different kernels are used to determine W_{ij} and its gradient ∇W_{ij} (*e.g.*, M. MÜLLER *et al.* [2003]). However, in a predictor-corrector method the same kernel must be used for both. Otherwise, the prediction and correction steps are not consistent. In the presented solver the cubic spline kernel according to the work of MONAGHAN [1992] was used.

The Gaussian is approximated in order to obtain a kernel function with compact support. This means the function vanishes at a finite distance which is also known as the support radius r. Such a kernel function can be written as $W_r(q(\mathbf{x}))$ with $q = \frac{\|\mathbf{x}\|}{r}$, where the function is non-zero for $0 \leq q < 1$. Therefore, the kernel gradient $\nabla W_r(q(\mathbf{x}))$ has the same compact support. As the kernel gradients must be determined for each particle in the neighborhood of another particle during the non-pressure force computation, the solver step iterations, and during the computation of the factors α_i and β_i, their evaluation is one of the most time consuming tasks in in the simulation. However, due to the large memory consumption it is not recommended to store the gradients of the particles when simulating large scenes. In the following, an efficient evaluation of the kernel and its gradient based on precomputed lookup tables is presented. Lookup tables have already been employed in other areas to speed up function evaluations but to the best of my knowledge they have not been used in existing SPH based simulations.

The generation of a lookup table for the kernel W_r is simple and can be realized by a regular sampling of the smooth scalar function with compact support. However, the gradient ∇W_r is a vectorial function and sampling this function in all three dimensions results in a higher memory consumption and computational overhead. To circumvent this issue a scalar function $g(q)$ can be extracted such that

$$\nabla W_r(q(\mathbf{x})) = \mathbf{x} \cdot g(q) \quad \text{with} \quad g(q) = \frac{\partial W_r}{\partial q} \cdot \frac{1}{r\|x\|}.$$

This function is also a smooth scalar function with compact support which can be sampled regularly to get a corresponding lookup table. Finally, only a single lookup and a multiplication with $\mathbf{x}$ is required to compute a kernel gradient. The kernel lookup tables are easy to implement and can also be used with other kernels and other SPH based simulation methods. Details about the sampling distance, the performance gain and the approximation error are discussed in Section 6.8.

6.8 Results

In this section simulation results using DFSPH are presented. All timings were measured on two Intel Xeon E5-2697 processors with 2.7 GHz, 12 cores per processor and 64 GB RAM. OpenMP was used to parallelize the code. Boundary handling and solid-fluid coupling was realized using the method ofAKINCI *et al.* [2012] and the neighborhood search using the parallel method of IHMSEN *et al.* [2011]. The method was moreover succesfully combined and tested with the surface tension model of BECKER and TESCHNER [2007], AKINCI *et al.* [2013a] and HE *et al.* [2014]. However, in the experiments presented in this paper only the method of AKINCI *et al.* [2013a] was used. In order to simulate viscosity, the XSPH variant of SCHECHTER and BRIDSON [2012] was employed.

A well-known issue of SPH fluid simulations is the problem of particle deficiencies at free surfaces. These lead to an underestimation of the density and therefore to particle clustering. To alleviate the problem, negative pressures were clamped to zero which is a common solution in computer graphics applications, see *e.g.*, [IHMSEN *et al.* 2014a]. In the performed simulations an average density error of less than $0.01\,\%$ and a density error due to the density change rate of less than $0.1\,\%$ were enforced. Moreover, adaptive time-stepping according to the CFL condition (*cf.* Section 6.4) was employed.

Performance

The performance of the novel simulation method for incompressible fluids was compared with IISPH, PBF and PCISPH using a breaking dam scenario with 125k particles with different fixed time step sizes. The particle radius in the simulation was $0.02\,\text{m}$. The results for a simulation over one second are summarized in Table 6.1 and Table 6.2 shows the according speedup factors. Note that IHMSEN *et al.* [2014a] compared the performance between PCISPH and IISPH with a similar scenario and measured comparable results.

DFSPH enforces the constant density condition and the divergence-free condition at the same time. Therefore, the density error is kept small during the whole simulation which leads to a significantly lower number of iterations required by the solvers. The iteration count is further reduced by a warm start which initializes both solvers with the sum of the stiffness values of the previous time step. In the dam break scenario a speedup factor of approximately 3 using the warm start was measured. While

	DFSPH			**IISPH**			**PBF**			**PCISPH**		
Δt [ms]	iter. (cd/df)	solver [s]	total [s]	iter.	solver [s]	total [s]	iter. [s]	solver [s]	total	iter. [s]	solver [s]	total
4.0	4.5/2.8	45.2	**51.3**	50.5	312.1	318.1	105.7	607.1	613.5	160.0	1079.1	1085.7
2.0	2.1/1.3	47.8	59.4	21.4	256.4	267.9	42.7	508.4	520.7	73.9	1021.2	1033.5
1.0	2.0/1.0	85.0	107.5	7.3	197.9	**220.7**	13.2	314.8	338.0	23.9	656.9	680.0
0.5	2.0/1.0	164.1	210.8	2.3	182.3	225.6	3.4	194.1	**240.5**	6.7	394.9	**440.9**
0.25	2.0/1.0	288.3	372.0	2.0	322.5	402.2	2.0	263.3	354.0	3.0	409.3	498.0

Table 6.1: Performance comparison of DFSPH, IISPH, PBF and PCISPH for a breaking dam simulation with 125k particles using different fixed time step sizes (see Figure 6.1). The table contains the average iteration count, the total computation time of the solvers and the total simulation time including the neighborhood search in a simulation over one second. The best total computation times are marked bold. The column with the average iteration count of DFSPH contains the values for the constant density solver (cd) and the divergence-free solver (df). Note that the solver time of DFSPH is the sum of the times needed by both solvers since they need almost the same amount of time for one iteration step.

	IISPH		**PBF**		**PCISPH**	
Δt [ms]	solver	total	solver	total	solver	total
4.0	6.9	6.2	13.4	12.0	23.9	21.2
2.0	5.3	4.5	10.6	8.8	21.4	17.4
1.0	2.3	2.1	3.7	3.1	7.7	6.3
0.5	1.1	1.1	1.2	1.1	2.4	2.1
0.25	1.1	1.1	0.9	1.0	1.4	1.3

Table 6.2: Speedup factors of DFSPH in comparison to IISPH, PBF and PCISPH based on the measurements in Table 6.1.

DFSPH uses a full warm start, IISPH performs best when multiplying the solution of the previous time step with a factor of 0.5 [IHMSEN *et al.* 2014a]. As summarized in Table 6.2 the performed warm start and the divergence-free velocity field in DFSPH lead to speedup factors of 6.9 in comparison to IISPH and up to 23.9 in comparison to PCISPH for a time step size of 4 ms. There are two reasons for this significant speedup. Firstly, in contrast to previous works, DFSPH enforces both incompressibility conditions which increases the stability of the simulation and therefore allows one to perform larger time steps and to apply a full warm start. Secondly, DFSPH reuses precomputed coefficients leading to faster iterations. The constant density solver step requires only an average of 4.5 iterations while the divergence-free solver step needed 2.8 iterations on average. The second best approach in the experiment was IISPH which already required 50.5 iterations. For smaller time step sizes DFSPH often used the minimum number of iterations which reduced the speedup for these step sizes. However, in simulations with more particles, where more iterations are required, the speedup is also larger for small step sizes. Based on the conducted experiments it can be noticed that DFSPH performs best when a time step size is chosen so that the number of iterations ranges between 2 and 20. The best performance of DFSPH method was obtained for larger time steps than IISPH, PBF and PCISPH (*cf.* Table 6.1). The usage of larger time steps has the advantage that the computationally expensive neighborhood search has to be executed less frequently.

In Section 6.7 a kernel optimization based on lookup tables was introduced. This optimization

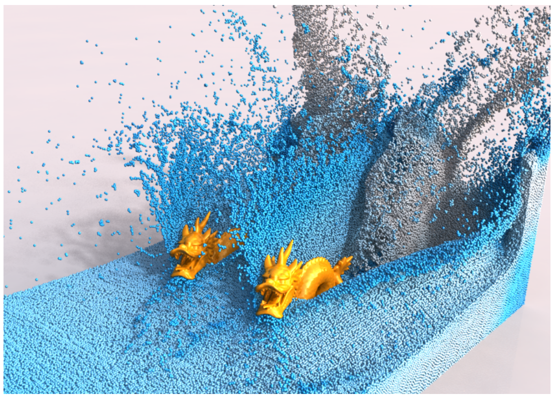


Figure 6.2: Breaking dam scenario with 2.3 million fluid particles and a complex boundary including two dragon models and a moving wall. The velocity field is linearly color coded: white represents the maximum and blue the minimum velocity.

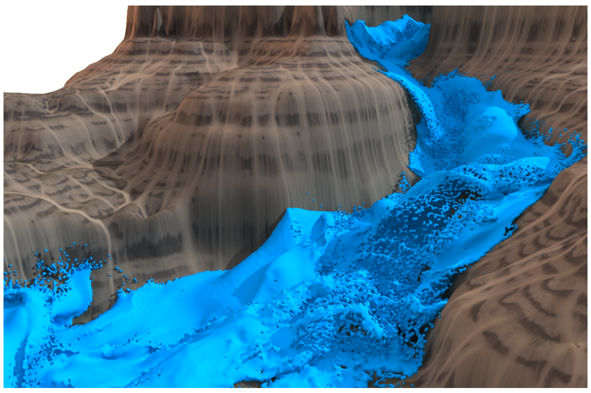


Figure 6.3: Whitewater flowing in a canyon. The scene contains 5 million fluid particles, 40 million boundary particles and the fluid is subject to a maximum volume compression of 0.01%.

Scenario	neigh. search	α	$\mathbf{F}^{\text{adv}}$	const. density	div.-free	**Total**
dragons	0.4	0.1	0.3	0.7	0.6	2.1
canyon	1.2	0.2	0.7	1.9	1.3	5.3

Table 6.3: Performance of the dam break (*cf.* Figure 6.2) and of the canyon simulation (*cf.* Figure 6.3). The table shows the average times per simulation step (in seconds) required by the neighborhood search, the computation of α, the computation of all non-pressure forces $\mathbf{F}^{\text{adv}}$, the constant density solver step and the divergence-free solver step.

yields a performance gain of approximately 30 % in the breaking dam scenario. In this performance test the kernel function and its gradient were sampled at 1000 points. For this sampling a maximum local error of less than $10^{-11}\,\text{m}^{-3}$ was measured. The implementation of the kernel optimization is very simple and the error is negligible, hence, the optimization is a useful extension for SPH simulations.

In order to demonstrate the performance of the presented method in more complex scenarios, another breaking dam simulation with a large number of fluid particles and a boundary consisting of two dragon models and a moving wall was performed (*cf.* Figure 6.2). In this scenario 2.3 million fluid particles and 0.7 million boundary particles were used. As a second example, 5 million fluid particles flowing through a canyon consisting of 40 million boundary particles were simulated (*cf.* Figure 6.3). Table 6.3 shows the average computation times of the most important steps in Algorithm 6.1 for both scenarios.

Memory Requirements

The memory requirements of the presented method are low. DFSPH only needs to store the scalar value α_i per particle which is then used in both solver steps. An additional scalar value has to be stored per particle for each solver step if a warm start is employed. Therefore, DFSPH requires considerably

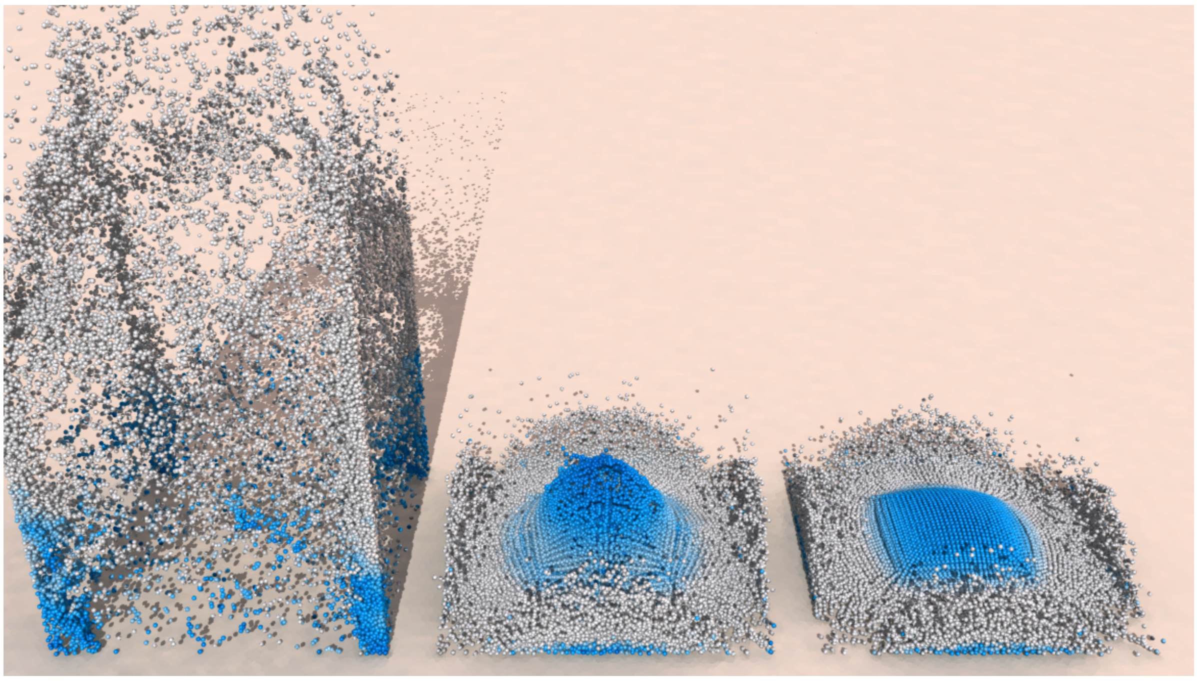


Figure 6.4: Stability comparison of DFSPH with PCISPH. The same color coding as in Figure 6.2 is used. A cube with 27k fluid particles is falling on the ground with a high velocity. Instabilities occur in the PCISPH simulation (left) and several fluid particles pass through the boundary. DFSPH without activating the divergence-free solver shows artifacts (middle) while DFSPH with activated divergence-free solver stays stable.

less memory than IISPH which needs seven scalar values for the solver and one for the warm start. This is especially an advantage when simulating large scale scenarios with millions of particles.

Stability

In this section it is demonstrated that current state-of-the-art SPH pressure solvers which do not enforce a divergence-free velocity field explicitly cannot satisfy the condition $\frac{D\rho}{Dt} = 0$ as demanded by the continuity equation. However, this can cause instabilities, especially when fluid particles are subject to high velocities.

The velocity divergence error of DFSPH and IISPH using a breaking dam scenario with 125k particles was compared (*cf.* Figure 6.1). The results demonstrate that DFSPH is able to maintain a divergence-free velocity field in contrast to approaches which do not enforce a divergence-free condition explicitly. In comparison the maximum local divergence error of IISPH was $108.3\,\mathrm{s}^{-1}$ while the error of DFSPH was only $1.9\,\mathrm{s}^{-1}$.

Large divergence errors can lead to instabilities as demonstrated in the following comparisons with PCISPH and IISPH. First, a cube of 27k fast moving particles falling on the ground was simulated in order to show how this influences the stability (*cf.* Figure 6.4). As previously mentioned, adaptive time-stepping driven by the CFL condition was used. In this experiment PCISPH, DFSPH with deactivated divergence-free solver step and DFSPH with activated divergence-free solver step were compared. A stable simulation was only possible with DFSPH when both solver steps were used. Without the

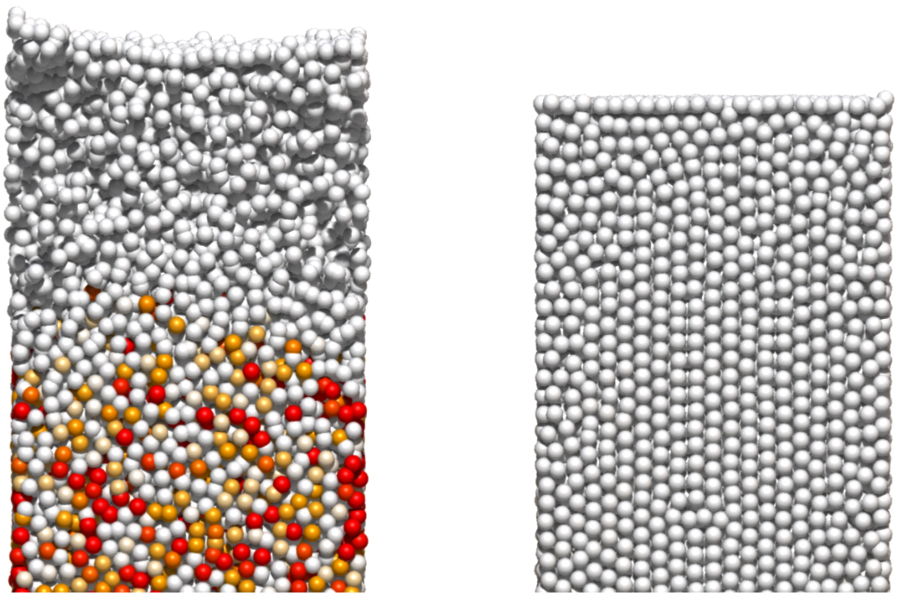


Figure 6.5: Top of a fluid pillar with 80k particles. The divergence errors are color coded: red is the maximum and white is the minimum error. Large divergence errors in the IISPH simulation (left) lead to jumping artifacts. The DFSPH approach (right) maintains a divergence-free velocity field which allows a stable simulation without artifacts.

divergence-free solver step artifacts occurred in the DFSPH simulation. PCISPH even became totally instable due to the impact and several fluid particles penetrated the boundary. A stable simulation with PCISPH was only possible when using a time step size which was clearly below the one suggested by the CFL condition which reduced the overall performance significantly.

To compare DFSPH with IISPH a resting fluid pillar model with 80k fluid particles was simulated (*cf.* Figure 6.5). Here, the time step size was restricted to a maximum of 5 ms because the CFL condition allows arbitrary large values for the resting particles. In the DFSPH simulation the maximum local divergence error was $2.5\,\mathrm{s}^{-1}$ which allowed a stable simulation without artifacts. In the IISPH simulation large divergence errors of up to $72.9\,\mathrm{s}^{-1}$ occurred. In combination with the large time steps this led to visual artifacts: fluid particles jumped up due to the errors (*cf.* Figure 6.5, left). By decreasing the time step size significantly this problem can be solved. However, this reduces the overall performance considerably. In summary, the comparisons show that enforcing a divergence-free velocity field improves the stability of the simulation significantly and allows to use larger time steps.

The next two experiments demonstrate the stability of DFSPH in simulations with dynamic boundaries and in large scale scenarios. In the first simulation several rigid bodies fall in a breaking dam scenario with 330k particles (*cf.* Figure 6.6). The Bullet physics library COUMANS [2015] was used to simulate the rigid bodies. In the second experiment 2.4 million fast moving particles hit a complex boundary (*cf.* Figure 6.7). In both scenarios DFSPH allowed a stable simulation with the maximum possible time step size guided by the CFL condition.

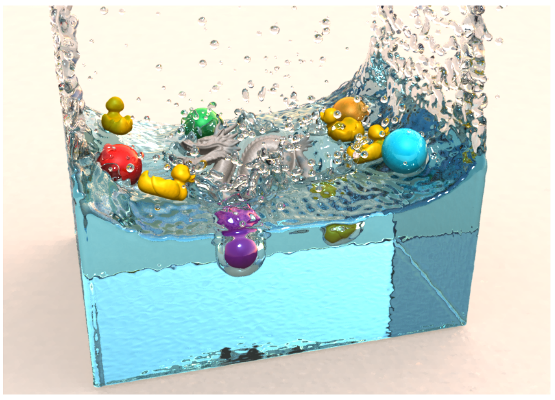


Figure 6.6: Two-way coupling of several dynamic rigid bodies with 330k fluid particles.

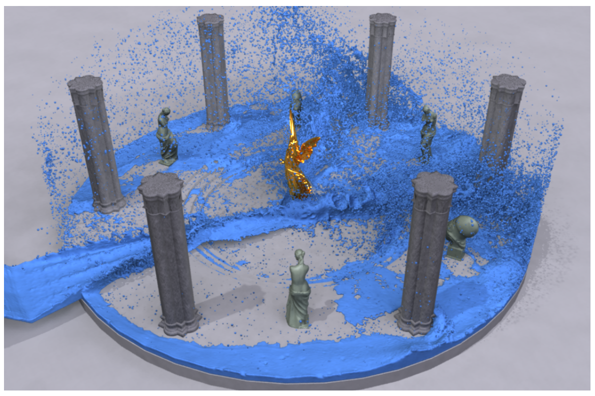


Figure 6.7: A temple is flooded with water. The scene contains 2.4 million fast moving fluid particles and a complex boundary consisting of 6 million particles.

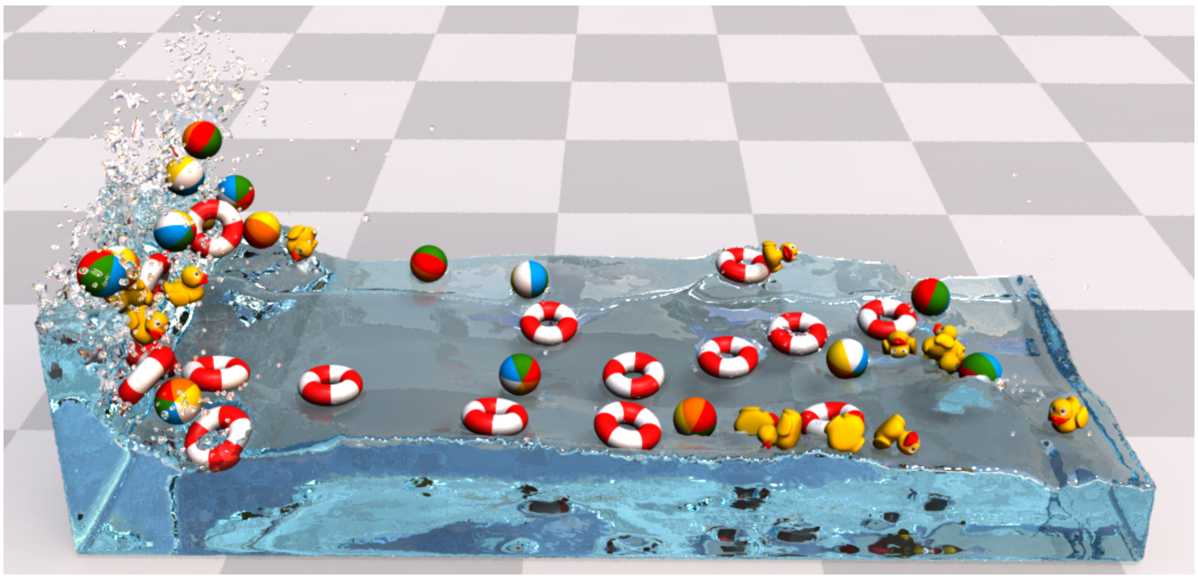
6.9 Conclusion

In this chapter a novel implicit SPH approach to simulate incompressible fluids was introduced. In contrast to previous state-of-the-art SPH methods for incompressible fluids the solver employs two distinct steps to enforce constant density and a divergence-free condition. Enforcing both conditions improves the stability and the performance of the simulation. The method handles scenarios with millions of fast moving particles robustly and is considerably faster than current state-of-the-art methods.

Since DFSPH is based on the same principles as other SPH pressure solvers, it has similar limitations. The density near free surfaces is underestimated due to particle deficiencies. This may lead to unnatural particle clustering artifacts. The problem was alleviated by clamping negative pressures to zero which is a common solution in computer graphics applications. However, a more sophisticated solution for this problem was presented by SCHECHTER and BRIDSON [2012]. They prevent particle deficiencies by generating ghost particles near the free surface to improve the physical behavior of the fluid. Avoiding pressure clamping by using ghost particles would also allow one to use linear equation system solvers with better convergence rates such as the conjugate gradient method.

Chapter 7

Density Maps for Improved SPH Boundary Handling



In this chapter, the novel concept of density maps for robust handling of static and rigid dynamic boundaries in fluid simulations based on SPH is presented. In contrast to the vast majority of existing approaches the presented method uses an implicit discretization for a continuous extension of the density field throughout solid boundaries. Using the novel representation accuracy of density and density gradient evaluations in boundary regions are enhanced and the computational efficiency is improved using efficient lookups into the density maps. A density map is generated in a preprocessing step and discretizes the density contribution in the boundary's near-field. In consequence of the high regularity of the continuous boundary density field, cubic Lagrange polynomials on a narrow-band structure of a regular grid are employed for discretization. This strategy not only removes the necessity to sample boundary surfaces with particles, but also decouples the particle size from the number of sample points required to represent the boundary. Moreover, it solves the ever-present problem of particle deficiencies near the boundary. Several comparisons presented in the following show that the representation is more accurate than particle samplings, especially for smooth curved boundaries. It is further demonstrated that the approach robustly handles scenarios with highly complex boundaries and even outperforms one of the most recent sampling based techniques.

7.1 Introduction

As discussed in the previous chapter, fluid simulations based on SPH have become increasingly popular in recent years. While a lot of work has been put into the development of novel models for the simulation of complex materials and efficient pressure solvers to enforce incompressibility, the general concept of discretizing boundary geometries using particle samplings remained largely untouched and is the state-of-the-art.

In this paper the concept of density maps, a novel model and discretization based on SDFs to represent boundary geometries and to impose boundary conditions with friction, is introduced. Starting from a continuous representation extending the fluid's density field into the boundary geometry, the function is discretized using cubic polynomials on a sparse grid without any dependence on particle sampling. The main advantage of the approach is that the higher-order approximation is able to very accurately represent curved boundary surfaces. This results in very smooth trajectories of particles in contact where traditional particle sampled boundaries produce artifacts in the dynamic movement as shown in the later presented results. Even in the rare case of particles strongly penetrating the boundary, our implicit representation throughout the boundary enables one to correctly recover non-penetration. An additional advantage of the approach is that the discretization can be precomputed such that the underlying integrals can be evaluated with high accuracy. Then, the discretized density map can be queried very efficiently during runtime using a simple interpolation function of constant complexity. Besides the efficiency improvement of the density lookups, the performance of neighborhood searches is greatly improved due to the vanished requirement to account for boundary particles. Additionally, the approach resolves the well-known problem of particle deficiencies in the boundary near-field while no particle sampling of the boundary is required. Furthermore, an approach for solid-fluid coupling with a novel Coulomb friction model is presented. As the density maps are based on SDFs, contact normals and friction tangents using the gradient of the map can be easily computed and the SDFs can even be reused for collision handling of dynamic objects.

The presented results show that the approach seamlessly integrates into existing SPH pressure solvers. On the basis of several comparisons with particle-based boundary representations, the advantages of the implicit representation are discussed. It will be further demonstrated that the novel formulation is able to produce physically plausible results in scenarios where particle sampled boundaries produce artifacts. The results, moreover, show that the method is able to robustly handle highly complex scenarios resulting in realistic dynamics.

7.2 Related Work

In recent years, a variety of SPH-based methods to simulate complex fluid phenomena and solid-fluid interaction has been developed. In this section, existing methods related to the presented approach will be discussed. For a general overview over SPH based fluid simulations I would like to refer the reader to the state-of-the-art report of IHMSEN *et al.* [2014b].

In an early work, MONAGHAN [1992] developed the first approach to simulate free surface flows with SPH using state equations. Later, DESBRUN and GASCUEL [1996] introduced the concept of SPH for the simulation of deformables and M. MÜLLER *et al.* [2003] for fluids to the computer graphics community. A spatially adaptive variant following the same approach was presented by B. ADAMS *et al.* [2007]. In order to simulate incompressible fluids, BECKER and TESCHNER [2007] proposed an equation of state based approach to guarantee a small maximum compression by precomputation of a suitable stiffness coefficient. A similar method for weakly compressible fluids based on a truly implicit time integration was later proposed by WEILER *et al.* [2016]. In the following years, many implicit solvers were developed to enforce incompressibility, either by demanding a divergence-free velocity field or by counteracting particle compression through position corrections or impulses. RAVEENDRAN *et al.* [2011] proposed a hybrid method using SPH and a background grid. While the fluid was represented by particles, a divergence-free velocity field was enforced on the grid. Similarly, a divergence-free velocity field was enforced without the requirement of a background grid using a meshless discretization of the pressure Poisson equation on the particles by HE *et al.* [2012a]. In contrast to enforce incompressibility, SOLENTHALER and PAJAROLA [2009] use a predictive-corrective scheme that avoids compression on the position level. A similar, position based approach was presented by MACKLIN and M. MÜLLER [2013]. Later, IHMSEN *et al.* [2014a] discretize the pressure Poisson equation to iteratively correct the density deviations, also enforcing incompressibility on the position level. TAKAHASHI *et al.* [2016] follow a similar concept proposing a hybrid SPH solver with interface handling for boundary conditions in the pressure Poisson equation. In contrast to either enforcing a divergence-free velocity field or counteracting compression based on particle positions, solvers satisfying both conditions were proposed, *e.g.*, by KANG and SAGONG [2014] and BENDER and KOSCHIER [2017]. In a recent work of DE GOES *et al.* [2015], a particle based method enforcing incompressibility based on Voronoi diagrams was presented.

Throughout the years, several works extending the existing solvers and modeling even more complex physical phenomena such as highly viscous materials [TAKAHASHI *et al.* 2015; PEER *et al.* 2015], multi-phase fluids and mixture models [SOLENTHALER and PAJAROLA 2008; REN *et al.* 2014] or surface tension and adhesion [AKINCI *et al.* 2013a] were proposed. Furthermore, a typical problem of traditional SPH discretizations is the particle deficiency near free surfaces. Therefore, SCHECHTER and BRIDSON [2012] presented an approach using ghost particles near the free surface while HE *et al.* [2014] introduced a method that modifies the internal pressure force and used an air pressure model to maintain stability throughout the simulation.

In order to handle static and dynamic solid boundaries traditional approaches sample the boundary surface using the concept of boundary particles [MONAGHAN 1994]. Furthermore, specialized boundary kernel functions are frequently used during the computation of boundary pressure forces as introduced by BECKER and TESCHNER [2007]. In contrast to counteracting boundary penetrations using forces, BECKER *et al.* [2009] proposed a predictor-corrector scheme for boundary handling. However, the method suffers from particle stacking artifacts near the boundary. In order to solve the problem, IHMSEN *et al.* [2010] presented a similar method that results in smoother density distributions by accounting for the fluid pressure near the boundary. A further improved variant of particle sampled boundaries for two-way coupling was proposed by AKINCI *et al.* [2012]. They additionally compute normalized pseudo masses for the boundary particles to account for non-uniform sampling patterns. A staggered particle approach was proposed by HE *et al.* [2012b]. They introduce auxiliary particles to enforce the desired boundary conditions during a pressure projection on a divergence-free velocity field on the staggered structure. Also adaptive sampling approaches for two-way coupling of fluids with deformable solids were developed over the years [M. MÜLLER *et al.* 2004b; YANG *et al.* 2012;

AKINCI *et al.* 2013b].

Eventually, all previously discussed approaches have one thing in common. They all discretize solid boundaries using particles which has the following consequences. Particle samplings of smooth surfaces are only moderately smooth and often introduce a certain noise due to the 'bumpy' patterns. These irregularities bias the dynamics of the fluid and cause an undesired friction effect which can also be interpreted as artificial viscosity. This effect will be demonstrated in the results section of this chapter. Furthermore, boundary particles produce a considerable computational overhead during neighborhood searches, force computations and pressure solves. In contrast, the novel concept of density maps is able to represent smooth surfaces very accurately using higher-order polynomials and improves the efficiency in neighborhood search algorithms and force computations.

Also approaches that avoid the use of particle samplings for boundary handling and two-way coupling were proposed. HARADA *et al.* [2007a] and HARADA *et al.* [2007b] present methods for static boundary handling. They construct a particle sampling for a planar example boundary and sample the boundary's density contribution dependent on the distance of a proxy fluid particle. They further construct an SDF for the boundary geometry and access the sampling using distance queries during runtime. However, the approach does not correctly account for non-planar boundaries. BODIN *et al.* [2012] developed a fluid solver that enforces boundary conditions using unilateral constraints. However, their approach eventually results in a *Mixed Linear Complementarity Problem* (MLCP) which is hard to solve and computationally intensive as discussed by ERLEBEN [2013]. In order to couple fluids with shells and cloth, HUBER *et al.* [2015] detect contacts using a continuous collision detection with the polygonal surfaces and use impulses and forces to counteract penetrations. In the field of mechanical engineering, KULASEGARAM *et al.* [2004] introduced a renormalization factor to improve the SPH approximation close to the boundary. The factor encodes the overlapping volume of kernel support and boundary domain. Similar gradient correction terms for weakly compressible fluids were introduced by FERRAND *et al.* [2013] for two-dimensional and by MAYRHOFER *et al.* [2015] for three-dimensional domains. The renormalization approach was later adopted by FUJISAWA and MIURA [2015] in the field of computer graphics. They represent boundaries using triangular meshes and compute the renormalization factors during runtime. While they avoid the requirement to sample the boundary geometry with particles, they report of a significant drop in performance due to the factor computation. Even worse, for larger numbers of triangles the computational effort increases resulting in a significantly worse performance compared to sampling approaches. A semi-analytic approach that extends the density field into the boundary was proposed by MONACO *et al.* [2011]. While they solve the radial portion of the occurring volume integrals analytically, they have to perform a numerical integration over the remaining dimensions during runtime. Very recently, a *Moving Least-Squares* (MLS) based approach was proposed by BAND *et al.* [2017]. They identify planar regions in a particle sampled boundary and fit a plane into the region using MLS. This improves the density estimate and normal computation resulting in a smoother representation. Unfortunately, their approach does not easily generalize to arbitrary boundary shapes.

In contrast to the discussed approaches, the here presented method precomputes an accurate implicit representation of an extended boundary density field for arbitrary boundary geometries. Consequently, the boundary density and density gradient contributing to a fluid particle can be efficiently accessed during runtime by performing only a single query into the density map.

7.3 Governing Equations and Standard SPH Discretization

As also discussed in the previous chapter, the dynamic behavior of a wide range of fluids can be modelled using the incompressible Navier-Stokes equations

$$\frac{D\rho}{Dt} = 0 \quad \Leftrightarrow \quad \nabla \cdot \mathbf{v} = 0 \tag{7.1}$$

$$\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{v} + \frac{\mathbf{f}}{\rho}, \tag{7.2}$$

where $\rho, t, \mathbf{v}, p, \nu$ and $\mathbf{f}$ stand for fluid density, time, velocity, pressure, kinematic viscosity and specific external forces, respectively. While the incompressibility condition is represented by Equation (7.1), Equation (7.2) describes momentum conservation. An analytical solution to these partial differential equations with arbitrary boundary conditions is unknown. Therefore, the equations have to be discretized in order to compute a numerical solution.

As already introduced in Chapter 6, a popular method for the spatial discretization of the problem is the SPH formalism. Following the standard approach scalar and vector fields are approximated and rewritten in discrete form as described in the following.

Let us recall the function smoothing of the prior to particle discretization of the SPH formalism (*cf.* 6.3) given a function $f : \Omega \to \mathbb{R}^n$:

$$f(\mathbf{x}) \approx \iiint\limits_{\mathcal{D}_\mathbf{x}} \frac{f(\mathbf{x}^*)}{\rho(\mathbf{x}^*)}\, W(\|\mathbf{x}-\mathbf{x}^*\|, h)\, \underbrace{\rho(\mathbf{x}^*)d\mathbf{x}^*}_{d\mathbf{m}}, \tag{7.3}$$

Following the standard approach this integral convolution is discretized into a particle sampling:

$$f(\mathbf{x}) \approx \sum_j \frac{m_j}{\rho_j} f(\mathbf{x}_j) W(\|\mathbf{x}-\mathbf{x}_j\|, h).$$

When the presented discretization is applied to the density field, the formula reduces to

$$\rho(\mathbf{x}) \approx \sum_j m_j W(\|\mathbf{x}-\mathbf{x}_j\|, h). \tag{7.4}$$

In order to account for solid boundaries the boundary surface is usually sampled using so-called boundary particles which serve as a continuous extension of the density field. Due to the field extension the density of a particle approaching the boundary will increase until the resulting pressure forces stop the particle from penetrating the boundary. Then Equation (7.4) is extended by a second sum over the contributing density of the boundary particles k:

$$\rho(\mathbf{x}) \approx \sum_j m_j W(\|\mathbf{x}-\mathbf{x}_j\|, h) + \sum_k \tilde{m}_k \tilde{W}(\|\mathbf{x}-\mathbf{x}_k\|, h).$$

Often specialized boundary kernels $\tilde{W}$ or special pseudo-masses $\tilde{m}_k$ are used as presented by BECKER and TESCHNER [2007] and AKINCI *et al.* [2012], respectively. Again, only boundary particles in the close neighborhood of the query point $\mathbf{x}$ are considered due to the compact supports of W and $\tilde{W}$.

Discussion Representing boundary regions with particle samplings has several disadvantages. The irregularity of the sampling patterns introduces a certain noise in the dynamics of the fluid. This also causes undesired friction between boundary and fluid which can be physically interpreted as artificial viscosity. These effects are discussed and illustrated in Section 7.6. In order to alleviate the effects of irregularities, the boundaries can be oversampled. This, however, results in a considerable computational overhead in neighborhood search algorithms and during pressure solves and force computations. Further, the detail with which smooth surfaces can be represented is dependent on the particle size so the number of required boundary particles increases when the size of the fluid particles is decreased.

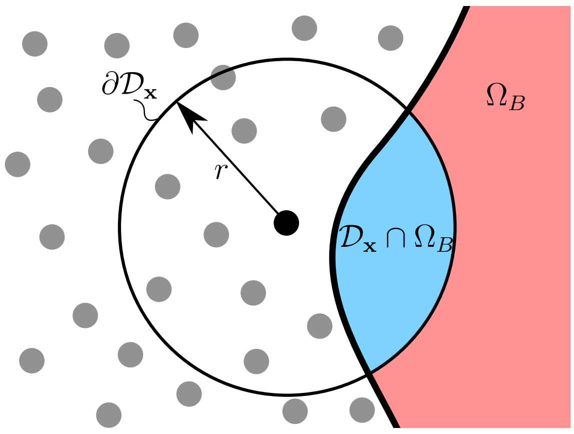

$\partial\mathcal{D}_{\mathbf{x}}$
r
Ω_B
$\mathcal{D}_{\mathbf{x}} \cap \Omega_B$

Figure 7.1: Domains required to generate the density map for a point located in a particle represented by a black dot. While the red area depicts the boundary domain Ω_B the blue area represents the integration domain for the evaluation of the boundary density value at the particle position.

7.4 Continuous Boundary Model

In this section the novel concept of density maps will be introduced. Recalling the continuous approximation described in Equation (7.3), the integral is additively decomposed by splitting the integration domain into boundary domain Ω_B and fluid domain $\mathcal{D}_x \setminus \Omega_B$ (*cf.* Figure 7.1) such that the density function can be rewritten as

$$\rho(\mathbf{x}) = \rho_F(\mathbf{x}) + \rho_B^c(\mathbf{x}),$$

where ρ_F denotes the density contribution of the fluid and ρ_B^c the density contribution of the boundary. The superscript c moreover denotes that were dealing with a continuous function prior to discretization. Subsequently, the integral is discretized over the fluid domain into the usual sum resulting in

$$\rho_F(\mathbf{x}) = \iiint\limits_{\mathcal{D}_x \setminus \Omega_B} W(\|\mathbf{x} - \mathbf{x}^*\|, h) d\mathbf{m} \approx \sum_j m_j W(\|\mathbf{x} - \mathbf{x}_j\|, h).$$

The basis for the boundary model is a virtual extension of the density field into the solid boundaries. Given a boundary object $\mathcal{B}$ the boundary density contribution in continuous form ρ_B^c is defined by

$$\rho_B^c(\mathbf{x}) = \iiint\limits_{\mathcal{D}_{\mathbf{x}} \cap \Omega_B} \gamma(\Phi(\mathbf{x}^*))\, W(\|\mathbf{x} - \mathbf{x}^*\|, h) d\mathbf{x}^* \tag{7.5}$$

$$\Phi(\mathbf{x}) = s(\mathbf{x}) \inf_{\mathbf{x}^* \in \partial\mathcal{B}} \|\mathbf{x} - \mathbf{x}^*\| \text{ with } s(\mathbf{x}) = \begin{cases} -1 & \text{if } \mathbf{x} \in \mathcal{B} \\ +1 & \text{otherwise} \end{cases}$$

$$\Omega_B = \operatorname{supp}(\gamma \circ \Phi) = \left\{ \mathbf{x} \in \mathbb{R}^3 \mid \gamma(\Phi(\mathbf{x})) \neq 0 \right\}.$$

Here $\gamma : \mathbb{R} \to \mathbb{R}^+$ is a (non-strictly) monotonically decreasing function which yields higher values for higher penetration depth. The signed distance represented by $\Phi : \mathbb{R}^3 \to \mathbb{R}$ is then passed as an argument to γ. From a physical point of view this means that if a particle approaches or penetrates the boundary surface, the basis density function γ yields increasing non-negative values. Finally, the resulting density field is modelled as convolution of γ with the SPH kernel W in order to fit the smoothed discretization. While a choice of γ as constant function is generally possible, the boundary's basis density field was modeled as continuous linear function

$$\gamma(x) = \begin{cases} \rho_0 \left(1 - \frac{x}{r}\right) & \text{if } x \leq r \\ 0 & \text{otherwise.} \end{cases} \tag{7.6}$$

The function's domain is chosen slightly larger than the boundary object $\mathcal{B}$ and equals the rest density ρ_0 on the boundary surface. The reason for choosing a linear function is twofold. Firstly, a piecewise constant choice of γ results in a discontinuity that makes the integral in Equation (7.5) much harder to evaluate numerically. Secondly, in the rare case that a particle strongly penetrates the boundary, the linear model enforces that the gradient direction of the function still points outwards in order to resolve the penetration. Obviously, the support of $\gamma(\Phi)$ on the domain Ω_B is enlarged by the kernel support radius r. The choice for defining γ on this extended boundary domain Ω_B rather than directly on $\mathcal{B}$ is motivated by practical considerations. As the particles are visualized using spheres or using a reconstructed polygonal surface with a certain offset, this buffer ensures that the fluid never visually penetrates the boundary surface. Also an interesting observation is that the sampling based boundary handling approach proposed by AKINCI *et al.* [2012] can be interpreted as particle discretization of the here presented model with $\gamma(s) = \rho_0$.

The final question that remains is: How can one accurately and efficiently compute the function values of ρ_B^c described by the integral in Equation (7.5)? Not only does this involve computing a numerical solution to an integral but also evaluating the signed distance to the boundary's surface. Generally, an evaluation during runtime, *e.g.*, using regular sampling, quadrature or Monte-Carlo sampling methods, is possible. However, an accurate numerical computation of the integral can be computationally costly as this may require a large number of function evaluations. Therefore, one either has to sacrifice accuracy for computational efficiency or vice versa.

A first step in the course of avoiding the runtime computation is the following important observation. For static boundaries ρ_B^c is, in terms of runtime variables, solely dependent on the query point $\mathbf{x}$. This also holds for rigid dynamic boundaries as the function can simply be evaluated at $\mathbf{R}^T(\mathbf{x} - \mathbf{c})$, where $\mathbf{R}$ and $\mathbf{c}$ represent the rotation of the body and its center of mass' translation, respectively. This means the correct density can be computed using a simple linear-affine transformation of the query point $\mathbf{x}$ into the rigid body's reference space and, thus, the contributing boundary density can be evaluated at any point in the boundary's reference space prior to the simulation. Consequently, this renders the possibility to precompute a highly accurate representation of the boundary density field. This will also improve the performance of the simulation due to the very efficient boundary density queries during runtime.

7.5 Discrete Density Maps

The purpose of this section is to develop a suitable discretization and construction algorithm for Equation (7.5). The suggestion to discretize the function using cubic polynomials of Serendipity type on a regular hexahedral grid is well justified for the following reasons. As discussed by M. JONES *et al.* [2006] signed distance functions are continuous everywhere. Further, the convolution of the boundary's basis density field $\gamma(\Phi(\mathbf{x}))$ with the (usually at least cubic polynomial) SPH kernel enforces a certain smoothness of ρ_B^c. The regularity of high-order polynomials is therefore well-suited to discretize the smooth function with high accuracy. An obvious choice for C^0 continuous polynomial discretization is a tensor-product space of Lagrange polynomials. Due to the regular nodal pattern implied by the basis, the approximation can be easily constructed by simply sampling the underlying field at the node pattern. However, in three dimensions a single cubic element requires 64 nodes compared to an 8-node linear element. In order to reduce the number of nodes, cubic Serendipity type elements can be employed. As discussed by ARNOLD and AWANOU [2011], this element type only requires 32 nodes and is able to maintain C^0 continuity while it has the same approximation order. A quantitative evaluation for an example boundary density field that demonstrates the superiority of the cubic Serendipity element compared to a sampling approach with trilinear interpolation is presented and discussed in Section 7.6. For a general introduction and discussion on Serendipity type elements for discretization, I would like to refer the reader to the works of ZIENKIEWICZ *et al.* [2013] and ARNOLD and AWANOU [2011].

As previously discussed, evaluations of the boundary density can be computationally intensive since they involve the evaluation of a signed distance to a possibly high-resolution polygonal surface and numerical integration to compute the convolution with the kernel function. In order to optimize evaluations the discretization process is split into three major steps:

1. Discretization of the signed distance function Φ,
2. Identification of grid cells close to the boundary resulting in a sparse, narrow-band pattern,
3. Discretization of the boundary density field ρ_B^c on the sparse grid.

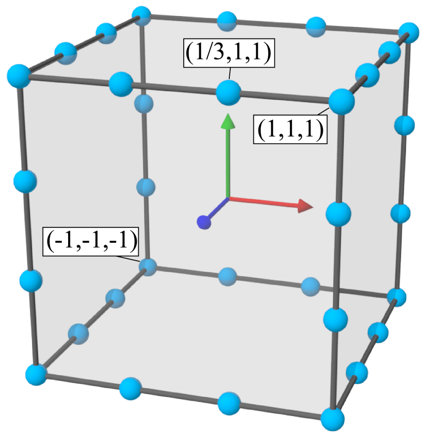


Figure 7.2: Cubic 32-node element of Serendipity type in isoparametric space.

I would like to stress the fact that all steps are part of the pre-processing leading to very efficient lookups in the course of the simulation. In the following paragraphs, each step will be described in detail.

Signed Distance Field Generation

In general, a signed distance field is either represented by analytic implicit functions or can be generated from explicit geometric representations. Bas-ed on the fact that complex geometries are usually represented by polygonal meshes, the algorithm to discretize the induced SDF using cubic Lagrange polynomials on a regular hexahedral grid will be described in the following.

Under the assumption that the input meshes are two-manifold, the unsigned distance to the mesh is computed and then a sign is determined using the pseudo-normal test proposed by BÆRENTZEN and AANÆS [2005]. In order to improve the performance of the signed distance computation, a bounding sphere hierarchy is constructed beforehand and traversed accordingly. As discussed earlier, isoparametric shape functions of Serendipity type with 32 nodes (*cf.* Figure 7.2) are employed and the exact signed distance is sampled on each node in world space. Then, the values $\Phi_i = \Phi(\mathbf{x}_i)$, corresponding to their sample points on the grid, can be used to cubically interpolate values within each cell using the according shape functions, *i.e.*,

- **Corner node** $(\xi_i = \pm 1,\, \eta_i = \pm 1,\, \zeta_i = \pm 1)$:
$$N_i = \frac{1}{64}(1+\xi_i\xi)(1+\eta_i\eta)(1+\zeta_i\zeta)\left[9(\xi^2+\eta^2+\zeta^2)-19\right] \tag{7.7}$$
- **Representative edge node** $(\xi_i = \pm\frac{1}{3},\, \eta_i = \pm 1,\, \zeta_i = \pm 1)$:
$$N_i = \frac{9}{64}(1-\xi^2)(1+9\xi_i\xi)(1+\eta_i\eta)(1+\zeta_i\zeta), \tag{7.8}$$

where $\boldsymbol{\xi}_i = (\xi_i, \eta_i, \zeta_i)^T$ represents the nodal positions in isoparametric space [ZIENKIEWICZ *et al.* 2013]. Here, Equation (7.8) describes the shape functions on a representative edge that is aligned in ξ-direction. The remaining edge nodes and according shape functions can easily be determined by permutation of coefficients ξ_i, η_i and ζ_i and coordinates ξ, η and ζ. Finally, the interpolation is described by

$$\Phi(\mathbf{x}) \approx \sum_{i=1}^{32} N_i(T(\mathbf{x}))\Phi_i$$
$$T(\mathbf{x}) = \left[\operatorname{diag}(\mathbf{b}-\mathbf{a})\right]^{-1}(2\mathbf{x}-(\mathbf{b}+\mathbf{a})),$$

where $\mathbf{a}$ and $\mathbf{b}$ are the hexahedral cell's minimum and maximum bound vectors and where $T : \mathbb{R}^3 \to \mathbb{R}^3$ maps a coordinate in world space to a coordinate in the element's isoparametric space.

Identification of Sparse Structures

As discussed earlier, the density field vanishes according to Equations (7.5) and (7.6) for signed distances of $\Phi > 2r$. Moreover, it can be assumed that particles will not penetrate the boundary more than three times their smoothing length as a heuristic. In fact, in none of the conducted experiments a particle penetrated the boundary at all. Consequently, the discretized distance field is used to identify unnecessarily stored cells. This greatly reduces the amount of memory required to store the discretized fields and avoids unnecessary interpolations in far away cells. Based on the previous observation and assumption all cells that do not meet the following criterion are discarded:

$$\begin{cases} \text{keep cell} & \text{if } 2r > \Phi_{\min} \,\wedge\, \Phi_{\max} > -r \\ \text{discard cell} & \text{otherwise,} \end{cases}$$

where Φ_{min} and Φ_{max} are the minimum and maximum signed distance from the cell to the boundary surface. Although no particles penetrated the boundary in all conducted experiments, the criterion was formulated in a way such that cells located within the radius r around the zero contour are kept as a safety buffer.

Boundary Density Field Evaluation and Discretization

In the third and final step, the boundary density function given by Equation (7.5) is evaluated at each node in the remaining sparse structure. As this is all part of the precomputation, the numerical integral evaluations can be performed with very high accuracy. In this work an adaptive Gauss-Kronrod quadrature algorithm was implemented that efficiently estimates the residual during computation using a nested quadrature rule [PIESSENS and BRANDERS 1974]. However, the usage of any quadrature method such as Monte-Carlo or naive sampling *etc.* is possible. After determination of all nodal coefficients, values at arbitrary positions can be interpolated within each cell using the functions described by Equations (7.7) and (7.8). A graphical illustration of a density map for the Stanford dragon is given in Figure 7.3. Please note that this image visualizes a 2D slice of a 3D density map. Therefore, also points located in close proximity to the surface in depth direction contribute to the field, *e.g.*, in the region around the tip of the dragons tail.

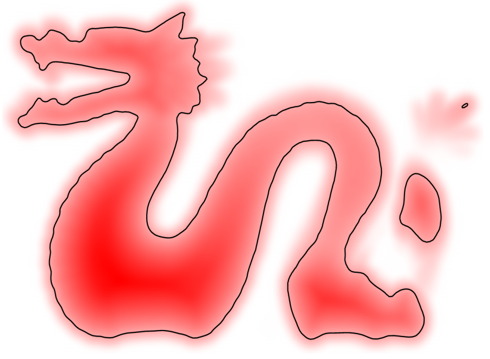


Figure 7.3: Two-dimensional slice of a three-dimensional density map. The boundary density contribution is visualized using a linear scale from white to red. The black contour represents the object surface intersecting the visualized slice.

Incorporation into Existing Pressure Solvers

Many types of fluids we encounter in daily life are nearly incompressible. Therefore, many implicit pressure solvers for the simulation of incompressible fluids were proposed in recent years, *e.g.*, [SOLENTHALER and PAJAROLA 2009; MACKLIN and M. MÜLLER 2013; IHMSEN *et al.* 2014a; BENDER and KOSCHIER 2017]. In this section it will be explained how the concept of density maps can be incorporated into existing solvers.

Density and Density Gradient

All pressure solvers typically have one thing in common. They all rely on density and density gradient evaluations in order to project the velocities onto a divergence-free field and/or to project the particle positions onto an uncompressed state. Both quantities can then be evaluated using

$$\begin{aligned}\rho_B^c(\mathbf{x}) \approx \rho_B(\mathbf{x}) &= \sum_{i=1}^{32} N_i(T(\mathbf{x}))\rho_{B,i} \\ \nabla_{\mathbf{x}}\rho_B^c\big|_{\mathbf{x}} \approx \nabla_{\mathbf{x}}\rho_B\big|_{\mathbf{x}} &= \sum_{i=1}^{32} \nabla N_i\big|_{\boldsymbol{\xi}=T(\mathbf{x})}\,\rho_{B,i},\end{aligned}$$

where $\rho_{B,i} = \rho_B^c(\boldsymbol{\xi}_i)$ are the boundary density values, evaluated at the grid nodes of the hexahedral cell containing $\mathbf{x}$.

Density Advection

The mentioned pressure solvers further rely on a density evaluation after an advection based on a candidate velocity field. Solvers of predictive-corrective nature, *e.g.*, [SOLENTHALER and PAJAROLA 2009; MACKLIN and M. MÜLLER 2013], actually perform the advection to get a candidate position for each particle and reevaluate the density. However, they often ignore the fact that the previously computed neighborhood information is invalidated by the advection and avoid recomputing the neighborhood information. Besides the problem that the neighborhood information for fluid particles is invalidated this also holds for boundary particles in boundary sampling based approaches. Fortunately, the here presented approach results in the exact result of the boundary density contribution after advections as the method does not rely on boundary particles.

Other pressure solvers [IHMSEN *et al.* 2014a; BENDER and KOSCHIER 2017] approximate the advected density at a particle i using a Taylor series of first order in time

$$\rho(\mathbf{x}_i(t+\Delta t)) \approx \rho(\mathbf{x}_i(t)) + \frac{D\rho(\mathbf{x}_i(t))}{Dt}\,\Delta t.$$

Generally, this strategy is very convenient as it circumvents the problem of invalidated neighborhood information at the expense of a loss in accuracy due to the first order approximation. However, when density maps are used, an actual advection of the boundary density contribution is both computationally cheap and very accurate. Therefore, the estimation is reformulated by splitting the density into fluid and boundary density

$$\begin{aligned}\rho(\mathbf{x}_i(t+\Delta t)) &= \rho_F(\mathbf{x}_i(t+\Delta t)) + \rho_B(\mathbf{x}_i(t+\Delta t)) \\ &\approx \rho_F(\mathbf{x}_i(t)) + \frac{D\rho_F(\mathbf{x}_i(t))}{Dt}\,\Delta t + \rho_B(\mathbf{x}_i(t+\Delta t)).\end{aligned}$$

When $\rho_B(\mathbf{x}_i(t+\Delta t))$ has to be evaluated for a particle i, the particle is simply advected in a prediction step and the density map is accessed yielding the desired result. If rigid dynamic boundaries are involved, one has also to account for their movement. In order to do so, the position and orientation of the body at time $t + \Delta t$ is computed in the prediction step by time integration prior to the evaluation of $\rho_B(\mathbf{x}_i(t+\Delta t))$.

Two-way Coupling with Dynamic Boundaries and Friction

In the course of the simulation, the fluid interacts with dynamic boundaries exerting forces on the fluid and vice-versa. While the influence of the rigid bodies on the fluid comes naturally due to their movement and spatially changing position of their underlying density maps, it must be additionally accounted for the forces exerted by the fluid on the rigid objects. Following Newton's third law, the negative force that the boundary exerted on the particle is simply applied on the dynamic boundary object in order to ensure conservation of momentum. Finally, the force a particle i exerts on a rigid object B at the point of contact is

$$\mathbf{f}_{B\leftarrow i} = m_i \frac{p_i}{\rho_i^2} \nabla_{\mathbf{x}_i} \rho_B|_{\mathbf{x}_i}.$$

Friction

Usually fluid friction is modeled as viscous force for energy dissipation near the boundary [BECKER and TESCHNER 2007; AKINCI *et al.* 2012]. In this case, however, the friction force is not necessarily acting tangential to the boundary surface. Therefore, a friction model inspired by Coulomb friction using the density map information is introduced in order to apply forces acting exactly tangential to the boundary surface. The friction force counteracting the slip of a particle i is then defined by

$$\mathbf{f}_{i,f} = -\mu\, m_i\, \frac{p_i}{\rho_i^2} \|\nabla_{\mathbf{x}_i} \rho_B\|\; \mathbf{t}_i$$

with tangent vector $\mathbf{t}_i = \Delta\mathbf{v}_i - (\Delta\mathbf{v}_i \cdot \mathbf{n}_i)\mathbf{n}_i$ and normal vector $\mathbf{n}_i = \nabla_{\mathbf{x}_i}\rho_B / \|\nabla_{\mathbf{x}_i}\rho_B\|$, where $\mu > 0$ is the friction coefficient and $\Delta\mathbf{v}_i = \mathbf{v}_i - \mathbf{v}_B$ is the relative velocity between the fluid particle and the boundary. For a dynamic rigid body with velocity $\mathbf{v}$ and angular velocity $\boldsymbol{\omega}$ the boundary velocity is determined by $\mathbf{v}_B = \mathbf{v} + (\boldsymbol{\omega} \times (\mathbf{x}_B - \mathbf{c}))$, where $\mathbf{x}_B$ denotes the point of contact. However, in order to avoid that a particle changes its tangential direction due to $\mathbf{f}_{i,f}$, the following definition of the friction force is used:

$$\mathbf{f}'_{i,f} = \begin{cases} \mathbf{f}_{i,f} & \text{if} \quad -\frac{\Delta t}{m_i}\mathbf{f}_{i,f} \cdot \mathbf{t}_i < \Delta\mathbf{v}_i \cdot \mathbf{t}_i \\ -\frac{m_i}{\Delta t}(\Delta\mathbf{v}_i \cdot \mathbf{t}_i)\mathbf{t}_i & \text{otherwise.} \end{cases}$$

Thanks to the implicit boundary representation, the density map's gradient to compute the force can be efficiently reused. Analogously to the application of pressure forces on the dynamic boundaries, the negative friction force is applied on the dynamic object in contact.

7.6 Results and Discussion

For the experiments in this section the novel boundary handling was integrated into two state-of-the-art SPH methods: DFSPH [BENDER and KOSCHIER 2017] and IISPH [IHMSEN *et al.* 2014a], and the the cubic spline kernel [MONAGHAN 1992] was used in all simulations. Following the suggestion of the corresponding authors an average compression of less than $0.01\,\%$ and a divergence-error of less than $0.1\,\%$ (only DFSPH) was enforced. Moreover, the parallel method of IHMSEN *et al.* [2011] to compute particle neighborhood information and XSPH to account for viscosity [SCHECHTER and BRIDSON 2012] was employed. For the simulation of rigid bodies the open-source library PositionBasedDynamics [BENDER 2017] was integrated into the simulator and the generated sparse SDFs were reused for the purpose of collision detection. All measurements in this section were performed on two Intel Xeon E5-2697 processors with 2.7 GHz, 12 cores per processor and 64GB RAM.

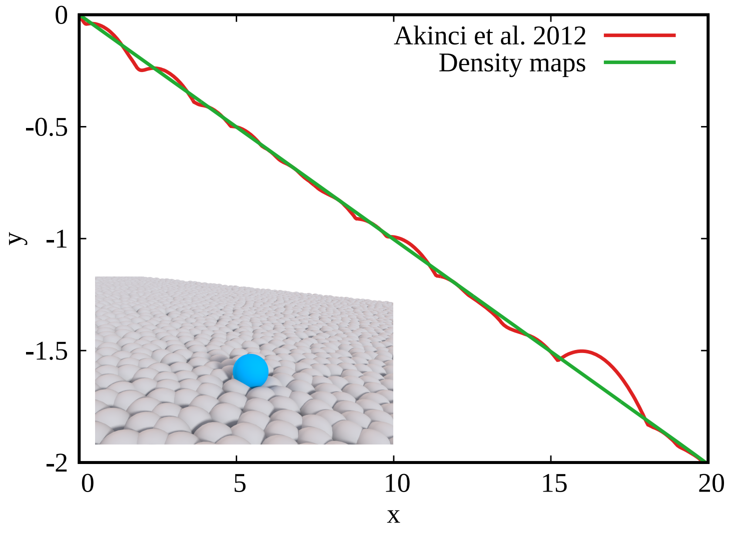


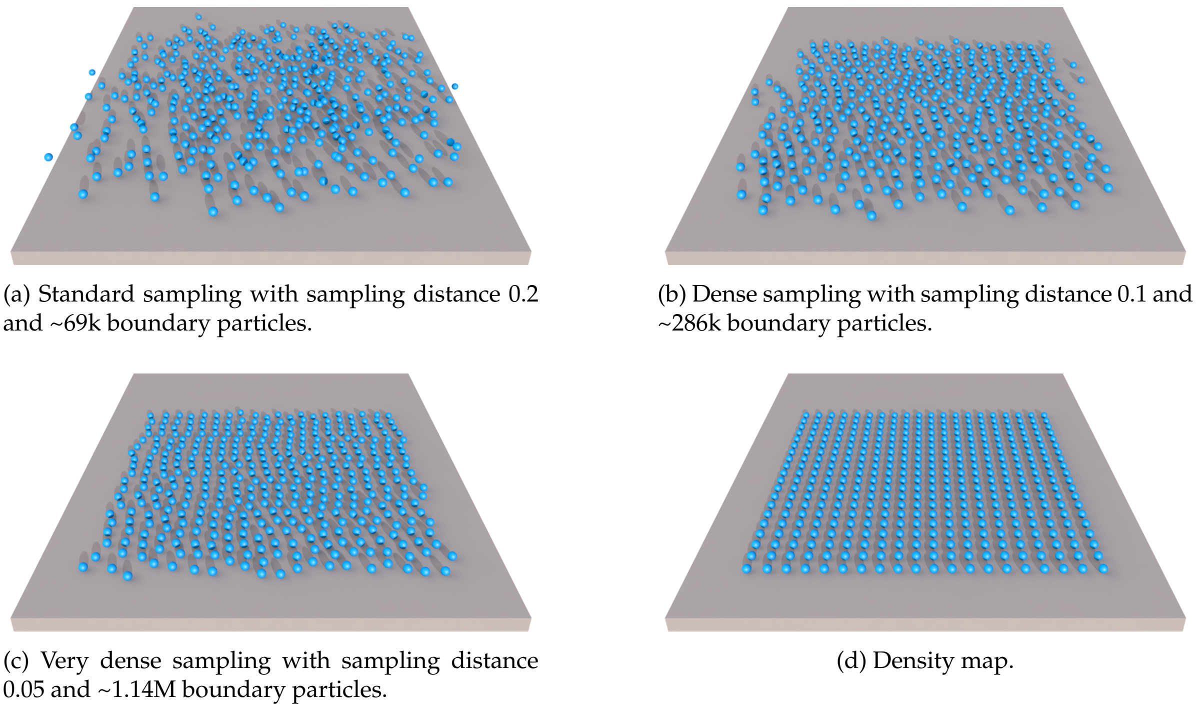
Figure 7.4: Trajectory of a single fluid particle flowing down an inclined plane. The used particle sampling with the sliding particle is depicted in the lower left corner.

Comparisons with Particle Sampled Boundaries

The behavior and performance of the novel boundary handling method was compared with the sampling based technique of AKINCI *et al.* [2012]. The first experiments demonstrate that density maps can accurately represent boundary surfaces while sampling based approaches may lead to artifacts and bias the resulting dynamics. The particle boundaries were generated using the Poisson-disk sampling algorithm with blue noise properties as described by CORSINI *et al.* [2012]. Naturally, other sampling algorithms can be used instead. Nevertheless, the general concept of particle sampling in three dimensions is inextricably linked to the irregularities in the boundary representation.

A single fluid particle was dropped on an inclined plane. Then, the particle trajectories resulting from boundary handling with the presented method and with the particle based boundary handling approach of AKINCI *et al.* [2012] were compared. The trajectories of the fluid particle are illustrated in Figure 7.4. The graphs clearly show that the particle jumps several times for the particle-sampled boundary representation while the here presented method lets the particle follow a smooth trajectory.

In a similar scenario depicted in Figure 7.5 an inclined plane was sampled in three different resolutions. While the boundary particles had a kernel support radius of $r = 0.8$ the planes were sampled with sampling distances 0.2, 0.1 and 0.05, respectively. The fourth plane was discretized using the density map approach with cell size 0.2. Then, a grid of fluid particles is dropped on each individual plane such that the particles do not influence each other due to the incompressibility condition being initially fulfilled. The trajectories of the fluid particles sliding on the sampled boundary are strongly biased resulting in a chaotic movement (*cf.* Figure 7.5a). In the case of a moderately or even strongly oversampled boundary the effect is alleviated. However, the irregularity of the sampling still considerably influences the particle dynamics (*cf.* Figures 7.5b and 7.5c). Finally, using the novel density map representation the particles slide down the slope as expected (*cf.* Figure 7.5d). I would further like to point out that dense samplings lead to a larger number of particles in each particle neighborhood. This again leads to a significant computational overhead in the neighborhood search.



(a) Standard sampling with sampling distance 0.2 and ~69k boundary particles.

(b) Dense sampling with sampling distance 0.1 and ~286k boundary particles.

(c) Very dense sampling with sampling distance 0.05 and ~1.14M boundary particles.

(d) Density map.

Figure 7.5: A grid of particles slides down an inclined plane. The plane was either discretized with particles or using the density map approach. The novel method maintains the grid structure while the boundary samplings bias the particle trajectories.

In a third and final comparison a fluid ($r = 0.2$) is trapped inside a spherical boundary with radius 3. Additionally, the boundary is rotating along the x-axis with constant angular velocity while the contact between fluid and boundary is frictionless. Again, a particle sampled boundary representation was compared with the density map approach with 100^3 grid cells. Due to the rotational invariance of the boundary geometry the rotation of the sphere should not influence the fluid. As illustrated in Figure 7.6 the density map perfectly fulfills this expectation while the fluid inside the particle based boundary never rests, because of oscillations induced by the uneven boundary. Even worse, sometimes particles started to jump due to the irregularity of the sampling.

Complex Scenarios

A breaking dam scenario using IISPH was simulated with 2.1 million particles in a box with rounded corners and 25 static Stanford armadillos as obstacles in the first scenario (*cf.* Figure 7.7). The density map of each armadillo was discretized by $100 \times 150 \times 100$ cells. However, the simulation required only one instance of the discretization as it was reused for all instances of the object. The result shows that the method is able to robustly handle large scenarios with millions of particles.

In the second scenario the performance of a fluid simulation using density maps was compared to the method of AKINCI *et al.* [2012] using DFSPH. A breaking dam scenario with a static Stanford dragon as obstacle was employed (*cf.* Figure 7.8). The fluid consisted of 211k particles while the obstacle was either sampled using 586k particles or discretized into $200 \times 160 \times 100$ cells. The average time step using the method of Akinci et al. took ~355ms of which the neighborhood search took ~57ms on

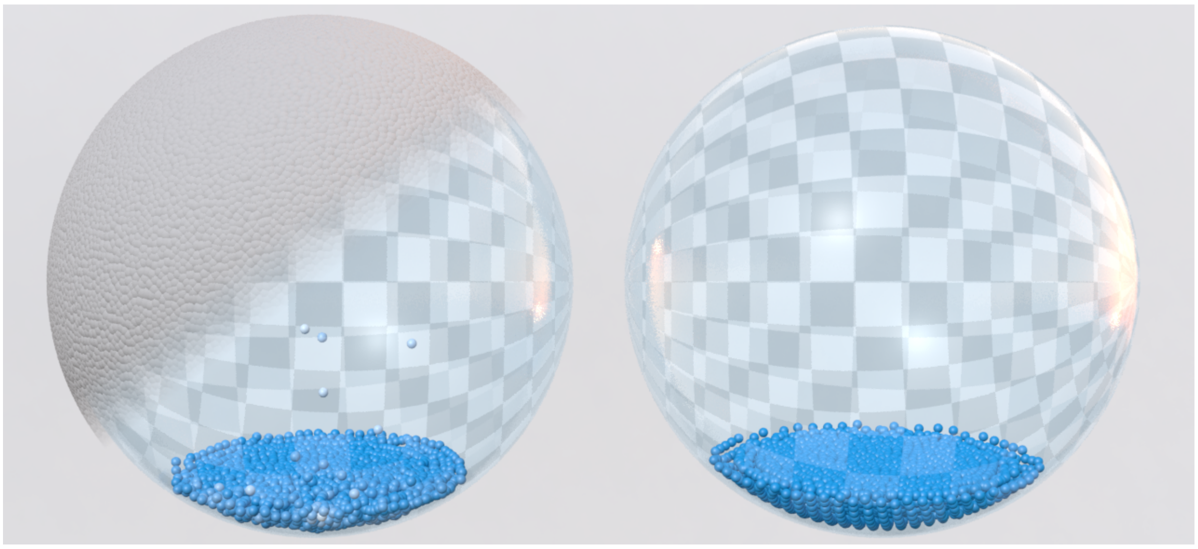


Figure 7.6: Comparison of frictionless fluid simulations inside rotating spheres with sampled (left) and implicit boundary (right). The sampling pattern is partially illustrated in the left image. Independent of the resolution, the particles never rest due to the irregularity of the boundary while the fluid handled by the implicit boundary rests perfectly.

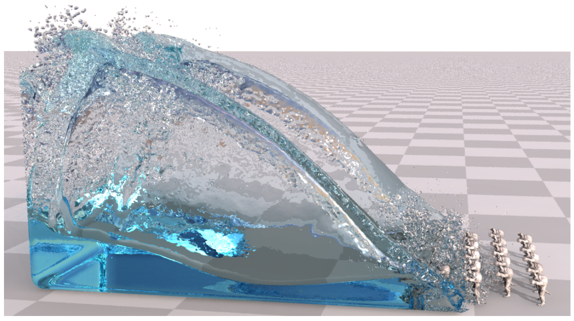


Figure 7.7: Breaking dam scenario consisting of 2.1 million fluid particles and 25 armadillos as static obstacles in a box.

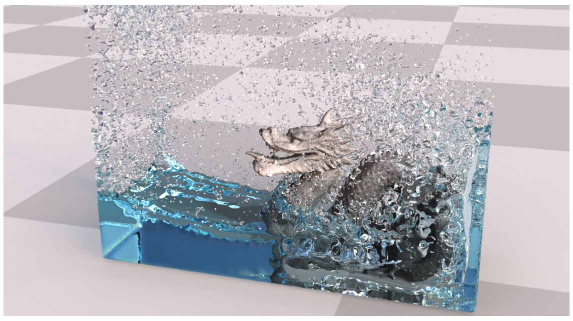


Figure 7.8: Benchmark scenario with a dragon obstacle.

average. A simulation using the novel method outperformed the previous one resulting in an average time step of ~241ms with ~33ms for the neighborhood search. On the basis of these results, a speed-up of factor approximately 1.5 was achieved. Moreover, the lack of boundary particles in the novel approach sped up the neighborhood search by a factor of approximately 2.

In a further experiment, three water wheels driven by 790k fluid particles using DFSPH were simulated where the wheels were discretized using $160 \times 160 \times 64$ cells (*cf.* Figure 7.9). The wheels were constrained using hinge joints resulting in a complex mechanical scenario. The result shows how the two-way coupling correctly accelerates the wheels resulting in a steady and realistic movement.

In the fourth scenario illustrated in Figure 7.10, a breaking dam scenario with 45 dynamic rigid objects of different shapes and 710k fluid particles was simulated using DFSPH without friction. The result demonstrates that the presented method greatly deals with several floating objects, interacting and colliding with each other while being in contact with the fluid.

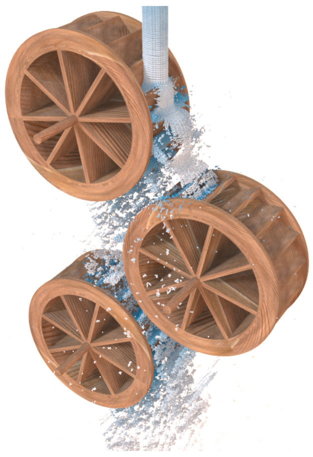


Figure 7.9: Three water wheels driven by 790k fluid particles.

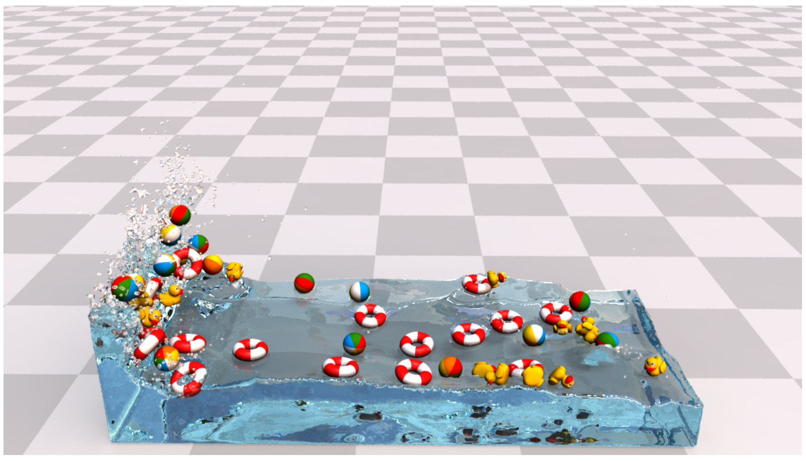


Figure 7.10: Breaking dam scenario with 45 dynamic rigid bodies interacting with 710k fluid particles.

The sixth experiment was simulated using IISPH and shows how three hollow spheres containing rubber ducks and fluids consisting of 510k particles in total with different friction coefficients of $\mu = 0$ (blue), $\mu = 250$ (green) and $\mu = 500$ (red) roll in a half-pipe. Please note that the same friction coefficient for contacts between the duck and the sphere was used in all three simulations. The blue fluid remains mostly on the ground of the sphere and does not react to its rotation. In contrast, the green fluid gets influenced by the inner surface of the hollow sphere following its rotation. The red fluid also shows the mentioned effect and even flips the rubber duck on its back after a few seconds. Another physically meaningful effect is that the rolling movement of the spheres containing the frictional fluids slows down earlier due to the energy dissipation.

In the final scene a ball floating on water ($\rho_0 = 1000$) was simulated. The ball initially had a density of 100. In the course of the simulation the density of the ball was doubled every three seconds. As illustrated in Figure 7.12 the ball initially lightly floats on the fluid surface. The gradually increased density lets the floating height decrease. When the density finally exceeds the rest density of the fluid, the ball sinks to the ground, as expected.

Linear vs. Cubic Density Maps To demonstrate the benefit of using cubic polynomials of Serendipity type instead of regular sampling with trilinear interpolation, a high-resolution reference solution for a circular boundary with radius 1 on a two-dimensional domain for particle support radius $r = 0.1$ was computed. Two discretizations of the function were generated. One using regular sampling and the other one using the cubic polynomials. Here, it is important to mention that approximately the same number of nodes was used in both grids. For the regular sampling a grid of 101×101 cells inheriting 10404 nodes was used while the cubic field was constructed for a grid of 45×45 cells with 10396 nodes. In order to evaluate the quality of the discretizations the distance to the reference solution relative to the reference density $\epsilon = |\rho_B - \rho_B^c|/\rho_0$ was determined. A plot of the error functions for both discretizations is depicted in Figure 7.13. Further, the average error for the regular sampling was 1.06×10^{-2}. The average error for the cubic field was 3.04×10^{-3} and therefore nearly one order of magnitude lower than the error of the linear discretization.

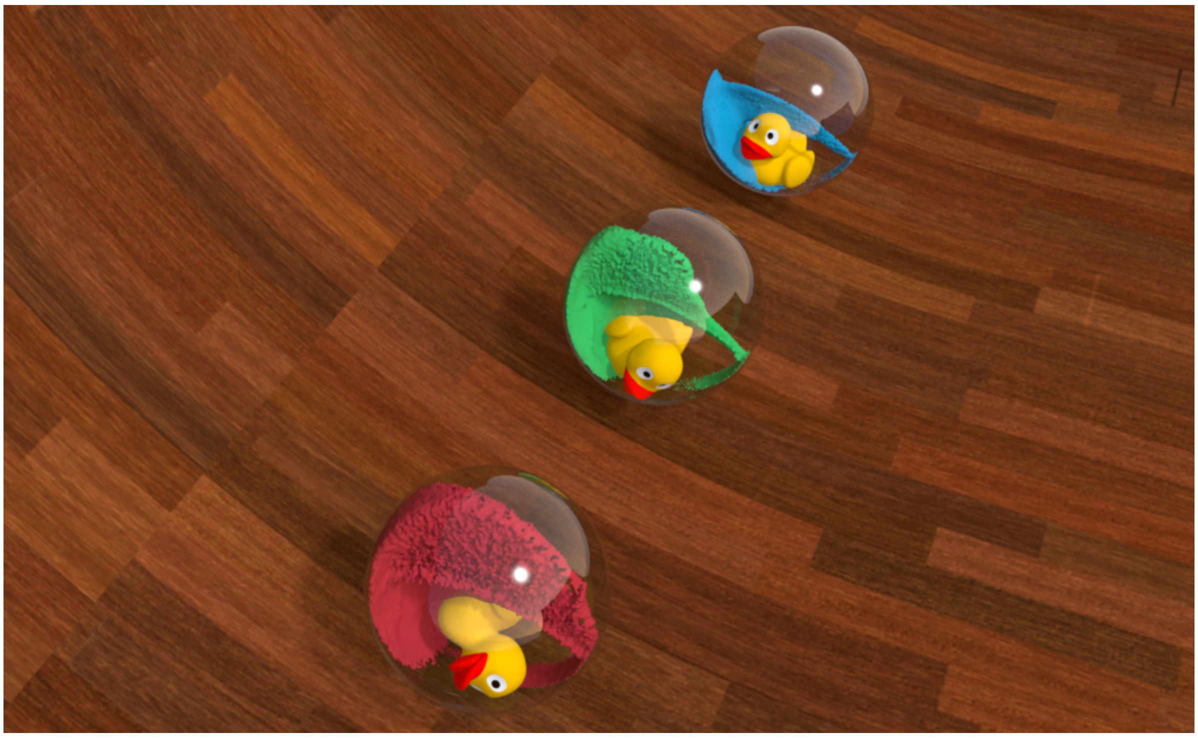


Figure 7.11: Hollow spheres with rubber ducks and 510k fluid particles are rolling in a half-pipe with different friction coefficients: $\mu = 0$ (blue), $\mu = 250$ (green) and $\mu = 500$ (red).

7.7 Conclusion

In this chapter the novel concept of density maps was presented; a method to simulate static and dynamic boundaries with friction in SPH simulations. Based on signed distance information, a continuous formulation that extends the fluid's density throughout the boundary was introduced. The function was then discretized using cubic polynomials on a sparse hexahedral grid in a precomputation step. This higher-order representation is able to very accurately represent curved surfaces. Even in cases of strong boundary penetrations, the involved particles can be correctly recovered. Moreover, the evaluation of the boundary density contribution using the interpolation function of the grid cells is computationally very efficient compared to approaches based on particle sampled boundaries. The presented results demonstrated that the method is more accurate and produces no artifacts compared to sampling based techniques. It also maintains stability even in highly complex scenarios and outperforms one of the most recent sampling based approaches.

Like every other method the approach has some limitations. Models with high frequency or sharp features are smoothed out due to the cubic polynomials. Therefore, using an adaptive discretization method in the spirit of [KOSCHIER *et al.* 2017b] could be beneficial. Another limitation is that the density maps cannot be precomputed for deformable objects, so further research on possibilites to perform updates is required. Also, no-slip boundary conditions were not considered in this work. However, these conditions can potentially be imposed by extending the velocity field into the boundary as described by MONACO *et al.* [2011].

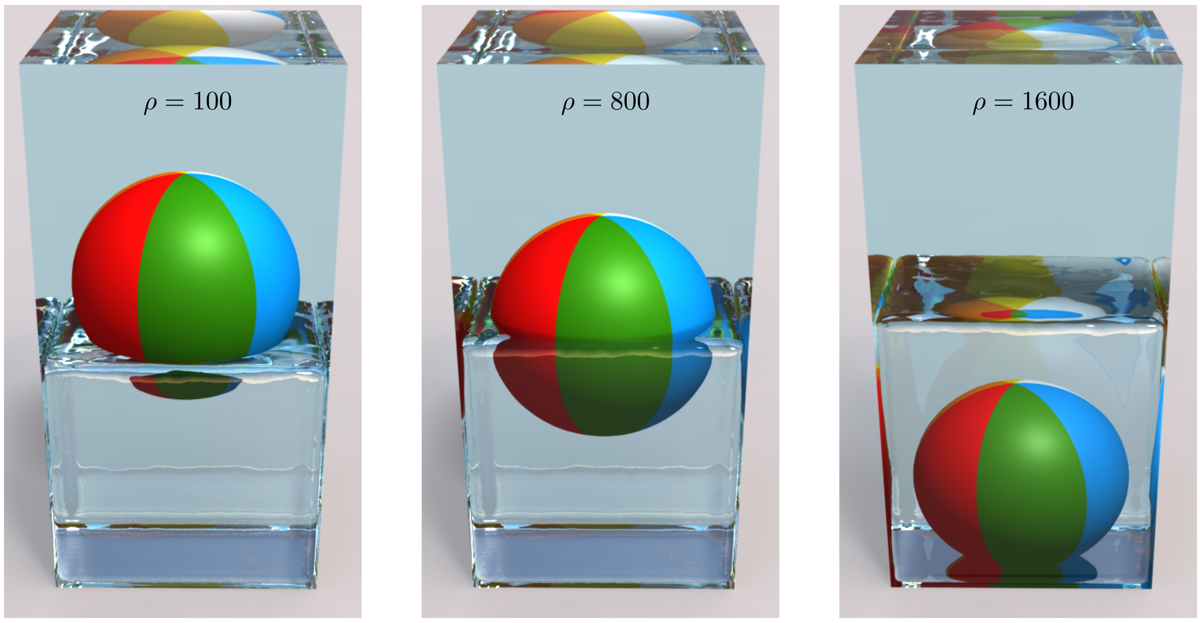


Figure 7.12: Solid ball with time-varying density interacts with water ($\rho_0 = 1000$).

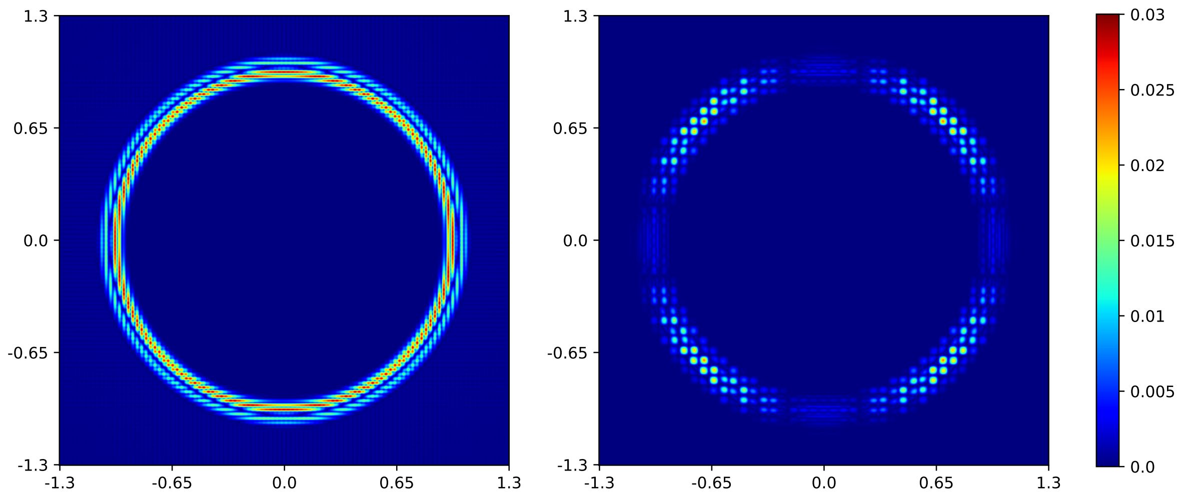


Figure 7.13: Error plots of piecewise linear (left) and cubic (right) density maps for a two-dimensional circular boundary with radius 1.0, smoothing length $h = 0.1$ and reference density $\rho_0 = 1000$. The colors indicate the error $\frac{|\rho_B - \rho_B^c|}{\rho_0}$ using the distance to the exact boundary density function ρ_B^c in relation to the reference density.

Chapter 8

Concluding Remarks

In this thesis, novel techniques for the physically based simulation of solids and fluids with special consideration of interfaces and boundaries were presented. This chapter summarizes the main contributions and provides an outlook for future research directions.

8.1 Summary

The numerical simulation of countless physical phenomena requires the representation and incorporation of interface and boundary geometries. These interface descriptions can represent boundary surfaces of solid objects, layers in composite materials, domain boundaries in fluid simulations, cuts, tears, and cracks in solids *etc*. Consequently, this thesis dealt with the numerical simulation of interface phenomena and was organized into three parts.

Cutting and Fracture

In Part I, techniques for the simulation of deformable solids with crack interfaces for modeling cutting and fracture were presented. Firstly, a novel method for the physically based simulation of brittle fracture using an adaptive tetrahedral discretization was proposed. The produced results demonstrate that the algorithm can produce realistic, detailed crack patterns, even when a coarse initial discretization is employed. The method includes a novel adaptive and reversible mesh refinement technique that allows to efficiently improve the quality of stress analyses in order to accurately locate crack origins. In this way, highly detailed cracks can be generated at moderate costs while additional refinement along the fracture surfaces produces visually appealing crack surfaces. Secondly, a novel approach for the simulation of complex cutting of three-dimensional deformable solids with fully implicit time integration was presented. Building on the concept of basis enrichment using the XFEM, deformable objects can be dissected without the requirement to alter the mesh. Following this strategy the explicitly represented cut surface can be captured implicitly in the discretization. In contrast to many existing approaches, even fine-structural cuts can be simulated resulting in a physically meaningful behavior by accurately accounting for integrals over discontinuous integrands while the number of additional degrees of freedom is kept to a minimum. The results demonstrate that the method is able to handle highly complex cut surfaces.

Implicit Boundary and Interface Representations

In Part II an adaptive higher-order representation for the discretization of complex signed distance functions describing boundary surfaces of volumetric objects was presented. Higher-order polynomials were efficiently fit to the input function by exploiting the properties of a shifted orthonormalized function basis. Then, the initial discretization defined on a regular grid was iteratively refined using spatial subdivision and using adaptions of the polynomial basis' degree. The conducted experiments showed that the proposed method greatly surpasses existing discretization approaches in accuracy while keeping the memory effort low. Moreover, it was demonstrated that the constructed discrete SDFs are well-suited for the purpose of collision detection in physically-based simulations; they are efficiently queryable and provide all required information to resolve a collision, *i.e.,* penetration depth and contact normal.

SPH Fluid Simulation with Implicit Boundary Handling

Part III presented methods to simulate incompressible fluids with a novel boundary handling technique using SPH discretizations. A novel pressure solver that enforces two incompressibility conditions was proposed. The first solver step ensures that a constant density field is maintained throughout the fluid. The second solver step further enforces a divergence-free velocity field. It was demonstrated that enforcing the two conditions improves both stability and performance of the simulation. Consequently, the method even outperforms the most recent implicit SPH pressure solvers. Besides the novel implicit pressure solver, an approach to handle non-penetration boundary conditions using the novel concept of density maps was presented. Based on the signed distance function of a given domain border, a continuous formulation that extends the fluid's density throughout the boundary was formulated. The function was then discretized using cubic polynomials on a sparse hexahedral grid in a precomputation step. This higher-order representation is able to very accurately represent curved surfaces. The presented results demonstrate that the method is more accurate and produces no artifacts compared to sampling based techniques. It moreover maintains stability even in highly complex scenarios and outperforms one of the most recent sampling based approaches.

8.2 Outlook

The presented thesis corroborates the important role of interface phenomena in the field of physics based simulation in computer graphics. Potential directions for future work in this topic are manifold and important open problems and ideas will be discussed in the following.

In the previous chapters, explicit and implicit representations for the geometry of interfaces and boundaries were presented. In many cases explicit representations in form of meshes are avoided in order to circumvent the complexity of generating high-quality meshes or remeshing. Therefore, people often rather opt for implicit representations on structured grids. While a promising approach for describing the geometry of a physical interface using an implicit adaptive higher-order representation on structured grids was presented in this thesis, further efforts should be made to improve discrete implicit representations. In order to accurately represent sharp features using state-of-the-art approaches, strong subdivision is inevitable to capture the features to sufficient accuracy. The reason for this requirement lies in the fact that smooth polynomials are not particularly well-suited to capture functions that are locally non-differentiable. It would therefore be highly beneficial to either find a better suited function basis for discretization or to store additional information on the grid in order to capture the function 'kinks' using a considerably smaller number of cells. Another problem is that the generation of implicit representations is often computationally intensive and for that reason cannot be performed during runtime in interactive applications. However, many applications require the interface geometry to change, *e.g.,* to represent the surface of deformable objects or to capture the evolution of fluid interfaces or free surfaces. The development of novel techniques to potentially perform local fast updates of precomputed discretizations would be beneficial.

Besides improving implicit interface representations, challenges concerning numerical methods for simulating interface phenomena that account for effects implied by a given interface have to be embraced. Especially, in finite element simulations interfaces are often used to describe the layer between two different interacting materials. This can be the layer separating two fluids, *e.g.,* water and air, two different solid materials, *e.g.,* skin and flesh, or a homogeneous material intersected by a cut. Due to the nature of the finite element discretizations, integrals over the given interface modeling a flux between each individual side or integrals over intersected cells to determine mass and stiffness matrices or certain field quantities have to be numerically evaluated. As this integrand arising in interface problems is often locally non-smooth or even singular, advanced techniques for the numerical evaluation of these integrals should be developed. A method to construct quadrature rules for the evaluation of integrals with piecewise continuous polynomial integrands to simulate cutting was presented in this thesis. Nevertheless, further development of methodologies for quadrature rule construction could open up a plethora of new possibilities to simulate even more complex physical phenomena or to drastically improve existing techniques in accuracy and/or performance. An interesting idea would be to move completely away from interface and boundary surface aligned discretizations. This would shift the complexity in the FE discretization process away from mesh generation towards numerical integral evaluation of intersected or partially filled cells. In this way arbitrarily shaped domains could be captured using (hierarchical) structured grids that are remarkably well-suited for implementations on highly parallel computer architectures, *e.g.,* multicore processors, clusters and GPUs. Moreover, the implementation of spatially adaptive approaches is more or less trivial on grid-like structures.

Finally, I would like to discuss a more general goal associated with physical simulation in the field of computer graphics. All methods presented in this thesis deal with the synthesis of physical or at least physically plausible motion, deformation, and interaction of solids and fluids. One can not necessarily expect to find quantitative (or even qualitative) compliance of simulations with real experiments. In contrast, the main goal of numerical simulations in mechanical engineering is to quantitatively analyze mechanical problems in order to accurately predict a certain behavior or to understand certain physical phenomena. There seems to be a gap between the two related research fields. While analyses in mechanical engineering often focus on computing highly accurate numerical solutions to very specific complicated problems to get an exhaustive understanding of the mechanical behavior, numerical methods in the field of computer graphics aim to provide efficient and extremely robust simulations modeling highly complex mechanical effects in order to create a believable but not necessarily physically accurate or even meaningful mechanical behavior. Methods for digital fabrication are on the rise and recent advances in capturing (non-rigid) motion of solids from videos are substantial. Therefore, there seems to be a high demand for simulation techniques targeted towards qualitative predictions and analyses of an object's mechanical behavior. An example would be to make qualitative predictions of the deformation behavior of fabricated objects in order to optimize fabricated structures for certain applications. Another example would be to use video captured deformations of a solid object in arbitrary scenarios in order to predict its behavior in other situations or to synthesize new motions with the same object by solving the inverse problem through numerical simulation. The development of techniques suitable for the numerical simulation of such problems that are robust enough to simulate general and highly complex scenarios, efficient enough to acquire results in acceptable time constraints, but sufficiently accurate to make qualitative statements regarding material properties and dynamic behavior, poses a great challenge for future research.

Bibliography

R. ABEYARATNE (2012). *Continuum Mechanics: Volume II of Lecture Notes on The Mechanics of Elastic Solids*. Tech. rep. MIT Department of Mechanical Engineering.

B. ADAMS, M. PAULY, R. KEISER, and L. J. GUIBAS (2007). "Adaptively Sampled Particle Fluids". *ACM Transactions on Graphics* 26.3, p. 48.

N. AKINCI, G. AKINCI, and M. TESCHNER (2013a). "Versatile surface tension and adhesion for SPH fluids". *ACM Transactions on Graphics* 32.6, pp. 1–8.

N. AKINCI, J. CORNELIS, G. AKINCI, and M. TESCHNER (May 2013b). "Coupling elastic solids with smoothed particle hydrodynamics fluids". *Computer Animation & Virtual Worlds* 24.3-4, pp. 195–203.

N. AKINCI, M. IHMSEN, G. AKINCI, B. SOLENTHALER, and M. TESCHNER (July 2012). "Versatile rigid-fluid coupling for incompressible SPH". *ACM Transactions on Graphics* 31.4, pp. 1–8.

J. ALLARD, F. FAURE, H. COURTECUISSE, F. FALIPOU, C. DURIEZ, and P. G. KRY (2010). "Volume Contact Constraints at Arbitrary Resolution". *ACM Transactions on Graphics* 29.4, 82:1–82:10.

R. ANDO, N. THUEREY, and C. WOJTAN (2013). "Highly adaptive liquid simulations on tetrahedral meshes". *ACM Transactions on Graphics* 32, 103:1–103:10.

P. M. AREIAS and T. BELYTSCHKO (Feb. 2006). "A comment on the article "A finite element method for simulation of strong and weak discontinuities in solid mechanics" by A. Hansbo and P. Hansbo [Comput. Methods Appl. Mech. Engrg. 193 (2004) 3523–3540]". *Computer Methods in Applied Mechanics and Engineering* 195.9-12, pp. 1275–1276.

D. N. ARNOLD and G. AWANOU (2011). "The Serendipity Family of Finite Elements". *Foundations of Computational Mathematics* 11.3, pp. 337–344.

I. BABUŠKA and M. SURI (1981). "Error estimates for the combined h and p versions of finite element method". *Numerische Mathematik* 37, pp. 252–277.

I. BABUŠKA, B. A. SZABO, and I. N. KATZ (1981). "The p-Version of the Finite Element Method". *SIAM Journal on Numerical Analysis* 18.3, pp. 515–545.

I. BABUŠKA and M. SURI (1994). "The p and h-p Versions of the Finite Element Method, Basic Principles and Properties". *SIAM Review* 36.4, pp. 578–632.

J. A. BÆRENTZEN and H. AANÆS (2005). "Signed distance computation using the angle weighted pseudonormal". *IEEE Transactions on Visualization & Computer Graphics* 11.3, pp. 243–253.

J. BAERENTZEN (2005). "Robust Generation of Signed Distance Fields from Triangle Meshes". *Volume Graphics*, pp. 167–175.

J. A. BÆRENTZEN (2002). "Manipulation of volumetric solids with applications to sculpting". Phd thesis. Technical University of Denmark, pp. 1–304.

S. BAND, C. GISSLER, and M. TESCHNER (2017). "Moving Least Squares Boundaries for SPH Fluids". *Proceedings of Virtual Reality Interactions & Physical Simulations (VRIPhys)*, pp. 1–8.

R. E. BANK, A. H. SHERMAN, and A. WEISER (1983). "Some Refinement Algorithms and Data Structures for Regular Local Mesh Refinement". *Scientific Computer*, pp. 3–17.

Z. BAO, J.-M. HONG, J. TERAN, and R. FEDKIW (2007). "Fracturing Rigid Materials". *IEEE Transactions on Visualization & Computer Graphics* 13.2, pp. 370–378.

D. BARAFF, A. WITKIN, and M. KASS (2003). "Untangling Cloth". *ACM Transactions on Graphics* 22.3, pp. 862–870.

J. BARBIČ and D. L. JAMES (2008). "Six-DoF Haptic Rendering of Contact Between Geometrically Complex Reduced Deformable Models". *IEEE Transactions on Haptics* 1.1, pp. 39–52.

M. Becker and M. Teschner (2007). "Weakly compressible SPH for free surface flows". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 1–8.

M. Becker, H. Tessendorf, and M. Teschner (May 2009). "Direct Forcing for Lagrangian Rigid-Fluid Coupling". *IEEE Transactions on Visualization & Computer Graphics* 15.3, pp. 493–503.

J. Bedrossian, J. H. von Brecht, S. Zhu, E. Sifakis, and J. M. Teran (2010). "A second order virtual node method for elliptic problems with interfaces and irregular domains". *Journal of Computational Physics* 229.18, pp. 6405–6426.

T. Belytschko and T. Black (1999). "Elastic crack growth in finite elements with minimal remeshing". *International Journal for Numerical Methods in Engineering* 45.5, pp. 601–620.

J. Bender (2017). *PositionBasedDynamics*. URL: github.com/InteractiveComputerGraphics/PositionBasedDynamics.

J. Bender, K. Erleben, and J. Trinkle (2014a). "Interactive Simulation of Rigid Body Dynamics in Computer Graphics". *Computer Graphics Forum* 33.1, pp. 246–270.

J. Bender and D. Koschier (2015). "Divergence-Free Smoothed Particle Hydrodynamics". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 1–9.

J. Bender and D. Koschier (2017). "Divergence-Free SPH for Incompressible and Viscous Fluids". *IEEE Transactions on Visualization & Computer Graphics* 23.3, pp. 1193–1206.

J. Bender, D. Koschier, P. Charrier, and D. Weber (2014b). "Position-Based Simulation of Continuous Materials". *Computers & Graphics* 44.1, pp. 1–10.

J. Bender, D. Koschier, T. Kugelstadt, and M. Weiler (July 2017). "A Micropolar Material Model for Turbulent SPH Fluids". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 1–8.

D. Bielser, P. Glardon, M. Teschner, and M. Gross (2004). "A State Machine for Real-Time Cutting of Tetrahedral Meshes". *Journal of Graphical Models* 66.6, pp. 398–417.

D. Bielser and M. H. Gross (2000). "Interactive Simulation of Surgical Cuts". *Pacific Graphics*, pp. 116–126.

D. Bielser, V. A. Maiwald, and M. H. Gross (1999). "Interactive Cuts through 3-Dimensional Soft Tissue". *Computer Graphics Forum* 18.3, pp. 31–38.

K. Bodin, C. Lacoursière, and M. Servin (2012). "Constraint fluids". *IEEE Transactions on Visualization & Computer Graphics* 18, pp. 516–526.

J. Bonet and R. D. Wood (1997). *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press.

R. Bridson (2008). *Fluid Simulation for Computer Graphics*. CRC Press.

R. Bridson, S. Marino, and R. Fedkiw (2003). "Simulation of Clothing with Folds and Wrinkles". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. SCA '03, pp. 28–36.

D. Burkhart, B. Hamann, and G. Umlauf (2010). "Adaptive and Feature-Preserving Subdivision for High-Quality Tetrahedral Meshes". *Computer Graphics Forum* 29.1, pp. 117–127.

O. Busaryev, T. K. Dey, and H. Wang (2013). "Adaptive Fracture Simulation of Multi-Layered Thin Plates". *ACM Transactions on Graphics* 32.4, p. 1.

F. Calakli and G. Taubin (2011). "SSD: Smooth Signed Distance Surface Reconstruction". *Computer Graphics Forum* 30.7, pp. 1993–2002.

S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popović (2002). "A Multiresolution Framework for Dynamic Deformations". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. SCA '02. San Antonio, Texas, pp. 41–47.

F. Chen, C. Wang, B. Xie, and H. Qin (2013). "Flexible and rapid animation of brittle fracture using the smoothed particle hydrodynamics formulation". *Computer Animation & Virtual Worlds*. Vol. 24, pp. 215–224.

A. J. Chorin (Oct. 1968). “Numerical Solution of the Navier-Stokes Equations”. *Mathematics of Computation* 22.104, p. 745.

M. Corsini, P. Cignoni, and R. Scopigno (June 2012). “Efficient and Flexible Sampling with Blue Noise Properties of Triangular Meshes”. *IEEE Transactions on Visualization & Computer Graphics* 18.6, pp. 914–924.

E. Coumans (2015). *Bullet*. Version 2.8. URL: https://bulletphysics.org/.

S. J. Cummins and M. Rudman (1999). “An SPH Projection Method”. *Journal of Computational Physics* 152, pp. 584–607.

C. Daux, N. Moes, J. Dolbow, N. Sukumar, and T. Belytschko (2000). “Arbitrary branched and intersecting cracks with the extended finite element method”. *International Journal for Numerical Methods in Engineering* 48.12, pp. 1741–1760.

F. de Goes, C. Wallez, J. Huang, D. Pavlov, and M. Desbrun (2015). “Power Particles: An incompressible fluid solver based on power diagrams”. *ACM Transactions on Graphics* 34.4, 50:1–50:11.

G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr (2001). “Dynamic Real-Time Deformations using Space & Time Adaptive Sampling”. *Conference on Computer Graphics & Interactive Techniques*. SIGGRAPH ’01. New York, NY, USA, pp. 31–36.

M. Desbrun and M.-P. Gascuel (1996). “Smoothed Particles: A new paradigm for animating highly deformable bodies”. *Eurographics Workshop on Computer Animation and Simulation*, pp. 61–76.

C. Deul, P. Charrier, and J. Bender (Sept. 2014). “Position-based rigid-body dynamics”. *Computer Animation & Virtual Worlds* 27.2, pp. 103–112.

C. Dick, J. Georgii, and R. Westermann (2010). “A Hexahedral Multigrid Approach for Simulating Cuts in Deformable Objects.” *IEEE Transactions on Visualization & Computer Graphics* 17.11, pp. 1663–1675.

K. Erleben (2013). “Numerical methods for linear complementarity problems in physics-based animation”. *ACM SIGGRAPH Courses*, pp. 1–42.

K. Erleben and H. Dohlmann (2008). “Signed Distance Fields Using Single-Pass GPU Scan Conversion of Tetrahedra”. *GPU Gems 3*. Addison-Wesley Professional. Chap. 34, pp. 741–763.

F. Faure, S. Barbier, J. Allard, and F. Falipou (2008). “Image-based Collision Detection and Response Between Arbitrary Volume Objects”. *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 155–162.

M. Ferrand, D. R. Laurence, B. D. Rogers, D. Violeau, and C. Kassiotis (Feb. 2013). “Unified semi-analytical wall boundary conditions for inviscid, laminar or turbulent flows in the meshless SPH method”. *International Journal for Numerical Methods in Fluids* 71.4, pp. 446–472.

N. Foster and R. Fedkiw (2001). “Practical Animation of Liquids”. *ACM Transactions on Graphics* 28, pp. 12–17.

T.-P. Fries and T. Belytschko (Aug. 2010). “The extended/generalized finite element method: An overview of the method and its applications”. *International Journal for Numerical Methods in Engineering* 84.3, pp. 253–304.

S. Frisken (2006). “Designing with distance fields”. *ACM SIGGRAPH Courses*, pp. 60–66.

S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones (2000). “Adaptively sampled distance fields: a general representation of shape for computer graphics”. *ACM Transactions on Graphics*, pp. 249–254.

A. Fuhrmann, G. Sobottka, and C. Gross (2003). “Distance Fields for Rapid Collision Detection in Physically Based Modeling”. *Computer Graphics and Vision*, pp. 1–8.

M. Fujisawa and K. T. Miura (2015). “An Efficient Boundary Handling with a Modified Density Calculation for SPH”. *Computer Graphics Forum* 34.7, pp. 155–162.

R. a. Gingold and J. Monaghan (1977). “Smoothed Particle Hydrodynamics: Theory and Application to Non-Spherical Stars”. *Monthly Notices of the Royal Astronomical Society* 181, pp. 375–389.

L. Glondu, M. Marchal, and G. Dumont (2013). “Real-Time Simulation of Brittle Fracture using Modal Analysis”. *IEEE Transactions on Visualization & Computer Graphics* 19.2, pp. 201–209.

L. GLONDU, S. C. SCHVARTZMAN, M. MARCHAL, G. DUMONT, and M. A. OTADUY (Jan. 2014). “Fast Collision Detection for Fracturing Rigid Bodies”. *IEEE Transactions on Visualization & Computer Graphics* 20.1, pp. 30–41.

O. GONZALEZ and A. M. STUART (2008). *A First Course in Continuum Mechanics*. Cambridge University Press.

E. GRINSPUN, P. KRYSL, and P. SCHRÖDER (July 2002). “CHARMS: A Simple Framework for Adaptive Simulation”. *ACM Transactions on Graphics*. SIGGRAPH ’02 21.3, pp. 281–290.

D. GROSS and T. SEELIG (2011). *Fracture Mechanics*. 2nd. Springer.

A. HANSBO and P. HANSBO (2004). “A finite element method for the simulation of strong and weak discontinuities in solid mechanics”. *Computer Methods in Applied Mechanics and Engineering* 193, pp. 3523–3540.

T. HARADA, S. KOSHIZUKA, and Y. KAWAGUCHI (2007a). “Smoothed particle hydrodynamics in complex shapes”. *Spring Conference on Computer Graphics*, pp. 191–197.

T. HARADA, S. KOSHIZUKA, and Y. KAWAGUCHI (2007b). “Smoothed Particle Hydrodynamics on GPUs”. *Computer Graphics International*, pp. 63–70.

X. HE, N. LIU, S. LI, H. WANG, and G. WANG (2012a). “Local Poisson SPH for Viscous Incompressible Fluids”. *Computer Graphics Forum* 31.

X. HE, N. LIU, G. WANG, F. ZHANG, S. LI, S. SHAO, and H. WANG (2012b). “Staggered meshless solid-fluid coupling”. *ACM Transactions on Graphics* 31.6, p. 1.

X. HE, H. WANG, F. ZHANG, H. WANG, G. WANG, and K. ZHOU (2014). “Robust Simulation of Sparsely Sampled Thin Features in SPH-Based Free Surface Flows”. *ACM Transactions on Graphics* 34.1, 7:1–7:9.

J. HEGEMANN, C. JIANG, C. SCHROEDER, and J. M. TERAN (2013). “A Level-Set Method for Ductile Fracture”. *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 193–201.

J. L. HELLRUNG, L. WANG, E. SIFAKIS, and J. M. TERAN (2012). “A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions”. *Journal of Computational Physics* 231.4, pp. 2015–2048.

D. J. HOLDYCH, D. R. NOBLE, and R. B. SECOR (2008). “Quadrature rules for triangular and tetrahedral elements with generalized functions”. *International Journal for Numerical Methods in Engineering* 73.9, pp. 1310–1327.

P. HOUSTON, B. SENIOR, and E. SÜLI (2003). “Sobolev Regularity Estimation for hp-Adaptive Finite Element Methods”. *Numerical Mathematics and Advanced Applications*. Springer Milan, pp. 631–656.

X. HU and N. ADAMS (2007). “An incompressible multi-phase SPH method”. *Journal of Computational Physics* 227, pp. 264–278.

J. H. J. HUANG, Y. L. Y. LI, R. CRAWFIS, S.-C. L. S.-C. LU, and S.-Y. L. S.-Y. LIOU (2001). “A complete distance field representation”. *Proceedings of Conference on Visualization*, pp. 1–8.

M. HUBER, B. EBERHARDT, and D. WEISKOPF (2015). “Boundary Handling at Cloth-Fluid Contact”. *Computer Graphics Forum* 34.1, pp. 14–25.

M. IHMSEN, N. AKINCI, M. BECKER, and M. TESCHNER (Mar. 2011). “A Parallel SPH Implementation on Multi-Core CPUs”. *Computer Graphics Forum* 30.1, pp. 99–112. arXiv: 1110.3711.

M. IHMSEN, N. AKINCI, M. GISSLER, and M. TESCHNER (2010). “Boundary handling and adaptive time-stepping for PCISPH”. *Proceedings of Virtual Reality Interactions & Physical Simulations (VRIPhys)*, pp. 79–88.

M. IHMSEN, J. CORNELIS, B. SOLENTHALER, C. HORVATH, and M. TESCHNER (2014a). “Implicit incompressible SPH”. *IEEE Transactions on Visualization & Computer Graphics* 20.3, pp. 426–435.

M. IHMSEN, J. ORTHMANN, B. SOLENTHALER, A. KOLB, and M. TESCHNER (2014b). “SPH Fluids in Computer Graphics”. *Eurographics (State of the Art Reports)*, pp. 21–42.

O. JAMRIŠKA (2010). “Interactive ray tracing of distance fields”. *Central European Seminar on Computer Graphics*. Vol. 2.

L. JEŘÁBKOVÁ and T. KUHLEN (2009). “Stable Cutting of Deformable Objects in Virtual Environments Using XFEM”. *IEEE Computer Graphics & Applications* 29.2, pp. 61–71.

K. L. JOHNSON (1985). *Contact Mechanics*. Cambridge University Press.

M. JONES, J. BAERENTZEN, and M. SRAMEK (July 2006). "3D distance fields: a survey of techniques and applications". *IEEE Transactions on Visualization & Computer Graphics* 12.4, pp. 581–599.

M. W. JONES (2004). "Distance field compression". *Journal of WSCG* 12.2, pp. 199–204.

T. JU, F. LOSASSO, S. SCHAEFER, and J. WARREN (2002). "Dual Contouring of Hermite Data". *ACM Transactions on Graphics* 21.3, pp. 339–346.

N. KANG and D. SAGONG (2014). "Incompressible SPH using the Divergence-Free Condition". *Computer Graphics Forum* 33.7, pp. 219–228.

D. M. KAUFMAN, S. SUEDA, and D. K. PAI (2007). "Contact trees: Adaptive Contact Sampling for Robust Dynamics". *ACM SIGGRAPH Sketches*.

P. KAUFMANN, S. MARTIN, M. BOTSCH, E. GRINSPUN, and M. GROSS (2009). "Enrichment Textures for Detailed Cutting of Shells". *ACM Transactions on Graphics* 28.3, 50:1–50:10.

P. KAUFMANN, S. MARTIN, M. BOTSCH, and M. GROSS (2008). "Flexible Simulation of Deformable Models Using Discontinuous Galerkin FEM". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 105–115.

J. KIM and N. S. POLLARD (2011). "Fast simulation of skeleton-driven deformable body characters". *ACM Transactions on Graphics* 30.5, pp. 1–19.

B. M. KLINGNER and J. R. SHEWCHUK (2008). "Aggressive Tetrahedral Mesh Improvement". *International Meshing Roundtable*, pp. 3–23.

L. KOBBELT (2000). "Sqrt(3)-Subdivision". *Conference on Computer Graphics & Interactive Techniques*. SIGGRAPH '00, pp. 103–112.

D. KOSCHIER and J. BENDER (July 2017). "Density Maps for Improved SPH Boundary Handling". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 1–10.

D. KOSCHIER, J. BENDER, and N. THUEREY (2017a). "Robust eXtended Finite Elements for Complex Cutting of Deformables". *ACM Transactions on Graphics* 36.4, 55:1–55:13.

D. KOSCHIER, C. DEUL, and J. BENDER (2016). "Hierarchical hp-Adaptive Signed Distance Fields". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 1–10.

D. KOSCHIER, C. DEUL, M. BRAND, and J. BENDER (2017b). "An hp-Adaptive Discretization Algorithm for Signed Distance Field Generation". *IEEE Transactions on Visualization & Computer Graphics* 23.10, pp. 2208–2221.

D. KOSCHIER, S. LIPPONER, and J. BENDER (2014). "Adaptive Tetrahedral Meshes for Brittle Fracture Simulation". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 1–10.

S. KULASEGARAM, J. BONET, R. W. LEWIS, and M. PROFIT (2004). "A variational formulation based contact algorithm for rigid boundaries in two-dimensional SPH applications". *Computational Mechanics* 33.4, pp. 316–325.

F. LABELLE and J. R. SHEWCHUK (July 2007). "Isosurface Stuffing : Fast Tetrahedral Meshes with Good Dihedral Angles". *ACM Transactions on Graphics*. SIGGRAPH '07 26.3.

M. LAI, E. KREMPL, and D. RUBEN (2009). *Introduction to Continuum Mechanics*. Butterworth-Heinemann, pp. 1–520.

J. A. LEVINE, A. W. BARGTEIL, C. CORSI, J. TESSENDORF, and R. GEIST (2014). "A Peridynamic Perspective on Spring-mass Fracture". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. SCA '14. Copenhagen, Denmark, pp. 47–55.

M. LIU and G. LIU (2010). "Smoothed Particle Hydrodynamics (SPH): an Overview and Recent Developments". *Archives of Computational Methods in Engineering* 17.1, pp. 25–76.

F. LOSASSO, J. O. TALTON, N. KWATRA, and R. FEDKIW (2008). "Two-way coupled SPH and particle level set fluid simulation". *IEEE Transactions on Visualization & Computer Graphics* 14, pp. 797–804.

G. Łukaszewicz (1999). *Micropolar Fluids*. Modeling and Simulation in Science, Engineering and Technology. Birkhäuser Boston.

M. Macklin and M. Müller (2013). "Position Based Fluids". *ACM Transactions on Graphics* 32.4, pp. 1–5.

P.-L. Manteaux, C. Wojtan, R. Narain, S. Redon, F. Faure, and M.-P. Cani (2017). "Adaptive Physically Based Models in Computer Graphics". *Computer Graphics Forum* 36.6, pp. 312–337.

S. Martin, P. Kaufmann, M. Botsch, M. Wicke, and M. Gross (2008). "Polyhedral Finite Elements Using Harmonic Basis Functions". *Computer Graphics Forum* 27.5, pp. 1521–1529.

A. Mayrhofer, M. Ferrand, C. Kassiotis, D. Violeau, and F.-X. Morel (Jan. 2015). "Unified semi-analytical wall boundary conditions in SPH: analytical extension to 3-D". *Numerical Algorithms* 68.1, pp. 15–34.

A. McAdams, Y. Zhu, A. Selle, M. Empey, R. Tamstorf, J. Teran, and E. Sifakis (2011). "Efficient elasticity for character skinning with contact and collisions". *ACM Transactions on Graphics* 30.4, 37:1–37:12.

N. Mitchell, M. Aanjaneya, R. Setaluri, and E. Sifakis (2015). "Non-manifold Level Sets: A multivalued implicit surface representation with applications to self-collision processing". *ACM Transactions on Graphics* 34.6, 247:1–247:9.

W. F. Mitchell and M. A. McClain (Oct. 2014). "A Comparison of hp -Adaptive Strategies for Elliptic Partial Differential Equations". *ACM Transactions on Mathematical Software* 41.1, pp. 1–39.

N. Moës, J. Dolbow, and T. Belytschko (1999). "A finite element method for crack growth without remeshing". *International Journal for Numerical Methods in Engineering* 46.1, pp. 131–150.

N. Molino, Z. Bao, and R. Fedkiw (2004). "A Virtual Node Algorithm for Changing Mesh Topology During Simulation". *ACM Transactions on Graphics* 23.3, pp. 385–392.

N. Molino, R. Bridson, J. Teran, and R. Fedkiw (2003). "A Crystalline, Red Green Strategy for Meshing Highly Deformable Objects with Tetrahedra". *International Meshing Roundtable*. IMR '03, pp. 103–114.

A. D. Monaco, S. Manenti, M. Gallati, S. Sibilla, G. Agate, and R. Guandalini (2011). "SPH Modeling of Solid Boundaries Through a Semi-Analytic Approach". *Engineering Applications of Computational Fluid Mechanics* 5.1, pp. 1–15.

J. Monaghan (1989). "On the problem of penetration in particle methods". *Journal of Computational Physics* 82.1, pp. 1–15.

J. Monaghan (1994). "Simulating Free Surface Flows with SPH". *Journal of Computational Physics* 110.2, pp. 399–406.

J. Monaghan (1992). "Smoothed Particle Hydrodynamics". *Annual Review of Astronomy and Astrophysics* 30.1, pp. 543–574. arXiv: arXiv:1007.1245v2.

S. E. Mousavi, E. Grinspun, and N. Sukumar (Aug. 2011a). "Harmonic enrichment functions: A unified treatment of multiple, intersecting and branched cracks in the extended finite element method". *International Journal for Numerical Methods in Engineering* 85.10, pp. 1306–1322.

S. E. Mousavi, E. Grinspun, and N. Sukumar (Apr. 2011b). "Higher-order extended finite elements with harmonic enrichment functions for complex crack problems". *International Journal for Numerical Methods in Engineering* 86.4-5, pp. 560–574.

K. Moustakas, D. Tzovaras, and M. G. Strintzis (2007). "SQ-Map: Efficient Layered Collision Detection and Haptic Rendering". *IEEE Transactions on Visualization & Computer Graphics* 13.1, pp. 80–93.

B. Müller, S. Krämer-Eis, F. Kummer, and M. Oberlack (2017). "A high-order discontinuous Galerkin method for compressible flows with immersed boundaries". *International Journal for Numerical Methods in Engineering* 110.1, pp. 3–30.

B. Müller, F. Kummer, and M. Oberlack (Nov. 2013). "Highly accurate surface and volume integration on implicit domains by means of moment-fitting". *International Journal for Numerical Methods in Engineering* 96.8, pp. 512–528.

B. MÜLLER, F. KUMMER, M. OBERLACK, and Y. WANG (Nov. 2012). "Simple multidimensional integration of discontinuous functions with application to level set methods". *International Journal for Numerical Methods in Engineering* 92.7, pp. 637–651.

M. MÜLLER, M. TESCHNER, and M. GROSS (2004a). "Physically-Based Simulation of Objects Represented by Surface Meshes". *Computer Graphics International*, pp. 26–33.

M. MÜLLER, D. CHARYPAR, and M. GROSS (2003). "Particle-Based Fluid Simulation for Interactive Applications". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 154–159.

M. MÜLLER, N. CHENTANEZ, and T.-Y. KIM (July 2013). "Real Time Dynamic Fracture with Volumetric Approximate Convex Decompositions". *ACM Transactions on Graphics*. SIGGRAPH '13 32.4, p. 1.

M. MÜLLER and M. GROSS (2004). "Interactive Virtual Materials". *Graphics Interface*, pp. 239–246.

M. MÜLLER, L. MCMILLAN, J. DORSEY, and R. JAGNOW (2001). "Real-time Simulation of Deformation and Fracture of Stiff Materials". *Proceedings of the Eurographic workshop on Computer animation and simulation*, pp. 113–124.

M. MÜLLER, S. SCHIRM, M. TESCHNER, B. HEIDELBERGER, and M. GROSS (July 2004b). "Interaction of fluids with deformable solids". *Computer Animation & Virtual Worlds* 15.34, pp. 159–171.

K. MUSETH (2013). "VDB: High-resolution Sparse Volumes with Dynamic Topology". *ACM Transactions on Graphics* 32.3, 27:1–27:22.

A. NORTON, G. TURK, B. BACON, J. GERTH, and P. SWEENEY (1991). "Animation of fracture by physical modeling". *The Visual Computer* 7.4, pp. 210–219.

J. F. O'BRIEN, A. W. BARGTEIL, and J. K. HODGINS (2002). "Graphical Modeling and Animation of Ductile Fracture". *ACM Transactions on Graphics* 21.3, pp. 291–294.

J. F. O'BRIEN and J. K. HODGINS (1999). "Graphical Modeling and Animation of Brittle Fracture". *In Proceedings of SIGGRAPH*. SIGGRAPH '99, pp. 137–146.

M. A. OTADUY, N. JAIN, A. SUD, and M. C. LIN (2004). "Haptic display of interaction between textured models". *IEEE Visualization*, pp. 297–304.

E. G. PARKER and J. F. O'BRIEN (2009). "Real-Time Deformation and Fracture in a Game Environment". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. SCA '09, pp. 165–175.

T. PATTERSON, N. MITCHELL, and E. SIFAKIS (2012). "Simulation of Complex Nonlinear Elastic Bodies using Lattice Deformers". *ACM Transactions on Graphics* 31.6, p. 1.

M. PAULY, R. KEISER, B. ADAMS, P. DUTRÉ, M. GROSS, and L. J. GUIBAS (2005). "Meshless Animation of Fracturing Solids". *ACM Transactions on Graphics* 24.3, pp. 957–964.

A. PEER, M. IHMSEN, J. CORNELIS, and M. TESCHNER (2015). "An Implicit Viscosity Formulation for SPH Fluids". *ACM Transactions on Graphics* 34.4, pp. 1–10.

R. N. PERRY and S. F. FRISKEN (2001). "Kizamu: A System for Sculpting Digital Characters". *Conference on Computer Graphics & Interactive Techniques*, pp. 47–56.

T. PFAFF, R. NARAIN, J. M. de JOYA, and J. F. O'BRIEN (July 2014). "Adaptive tearing and cracking of thin sheets". *ACM Transactions on Graphics* 33.4, pp. 1–9.

R. PIESSENS and M. BRANDERS (1974). "A Note on the Optimal Addition of Abscissas to Quadrature Formulas of Gauss and Lobatto". *Mathematics of Computation* 28.125, pp. 135–139.

H. QU, N. ZHANG, R. SHAO, A. KAUFMAN, and K. MUELLER (2004). "Feature Preserving Distance Fields". *IEEE Volume Visualization and Graphics*, pp. 39–46.

A. RAOULT (1986). "Non-polyconvexity of the stored energy function of a Saint Venant-Kirchhoff material". eng. *Aplikace matematiky* 31.6, pp. 417–419.

K. RAVEENDRAN, C. WOJTAN, and G. TURK (2011). "Hybrid smoothed particle hydrodynamics". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 33–42.

B. Ren, C. Li, X. Yan, M. C. Lin, J. Bonet, and S.-M. Hu (2014). "Multiple-Fluid SPH Simulation Using a Mixture Model". *ACM Transactions on Graphics* 33.5, pp. 1–11.

C. L. Richardson, J. Hegemann, E. Sifakis, J. Hellrung, and J. M. Teran (2011). "An XFEM method for modeling geometrically elaborate crack propagation in brittle materials". *International Journal for Numerical Methods in Engineering* 88.10, pp. 1042–1065.

A. Rosenfeld and J. L. Pfaltz (1966). "Sequential Operations in Digital Picture Processing". *Journal of the ACM* 13.4, pp. 471–494.

M. Sanchez, O. Fryazinov, and A. Pasko (2012). "Efficient Evaluation of Continuous Signed Distance to a Polygonal Mesh". *Spring Conference on Computer Graphics*, pp. 101–108.

H. Schechter and R. Bridson (2012). "Ghost SPH for animating water". *ACM Transactions on Graphics* 31.4, 61:1–61:8.

A. Schmidt and K. G. Siebert (2000). "A posteriori estimators for the h-p version of the finite element method in 1D". *Applied Numerical Mathematics* 35.1, pp. 43–66.

C. Schroeder, A. Stomakhin, R. Howes, and J. M. Teran (2014). "A second order virtual node algorithm for Navier-Stokes flow problems with interfacial forces and discontinuous material properties". *Journal of Computational Physics* 265, pp. 221–245.

J. R. Shewchuk (1994). *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Tech. rep. University of California at Berkeley.

J. R. Shewchuk (2002). *What Is a Good Linear Finite Element? Interpolation, Conditioning, Anisotropy, and Quality Measures*. Tech. rep. University of California at Berkeley.

E. Sifakis, K. G. Der, and R. Fedkiw (2007). "Arbitrary Cutting of Deformable Tetrahedralized Objects". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 73–80.

F. Sin, A. W. Bargteil, and J. K. Hodgins (2009). "A point-based method for animating incompressible flow". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, p. 247.

J. Smith, A. Witkin, and D. Baraff (2001). "Fast and Controllable Simulation of the Shattering of Brittle Objects". *Computer Graphics Forum* 20.2, pp. 81–90.

B. Solenthaler and R. Pajarola (2009). "Predictive-corrective incompressible SPH". *ACM Transactions on Graphics* 28.3, 40:1–40:6.

B. Solenthaler and R. Pajarola (2008). "Density Contrast SPH Interfaces". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 211–218.

P. Sonneveld (1989). "CGS, a fast Lanczos-type solver for nonsymmetric linear systems". *SIAM Journal on Scientific and Statistical Computing* 10.1, pp. 36–52.

D. Steinemann, M. Harders, M. Gross, and G. Szekely (2006). "Hybrid Cutting of Deformable Solids". *IEEE Virtual Reality Conference*, pp. 35–42.

P. D. Stolfo, A. Schröder, N. Zander, and S. Kollmannsberger (2016). "An easy treatment of hanging nodes in hp-finite elements". *Finite Elements in Analysis and Design* 121, pp. 101–117.

J. Su, C. Schroeder, and R. Fedkiw (2009). "Energy Stability and Fracture for Frame Rate Rigid Body Simulations". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. SCA '09, pp. 155–164.

T. Takahashi, Y. Dobashi, I. Fujishiro, T. Nishita, and M. Lin (2015). "Implicit Formulation for SPH-based Viscous Fluids". *Computer Graphics Forum* 34.2, pp. 493–502.

T. Takahashi, Y. Dobashi, T. Nishita, and M. Lin (2016). "An Efficient Hybrid Incompressible SPH Solver with Interface Handling for Boundary Conditions". *Computer Graphics Forum* to appear, pp. 1–13.

D. Terzopoulos and K. Fleischer (1988). "Modeling Inelastic Deformation: Viscoelasticity, Plasticity, Fracture". *Conference on Computer Graphics & Interactive Techniques*. Vol. 22. SIGGRAPH '88 4, pp. 269–278.

C. Tsakmakis (2013). *Lecture notes on Kontinuumsmechanik I*. TU Darmstadt.

G. VENTURA (2006). "On the elimination of quadrature subcells for discontinuous functions in the eXtended Finite-Element Method". *International Journal for Numerical Methods in Engineering* 66.5, pp. 761–795.

B. WANG, F. FAURE, and D. K. PAI (July 2012). "Adaptive image-based intersection volume". *ACM Transactions on Graphics* 31.4, pp. 1–9.

Y. WANG, C. JIANG, C. SCHROEDER, and J. TERAN (2014). "An Adaptive Virtual Node Algorithm with Robust Mesh Cutting". *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 1–9.

M. WEILER, D. KOSCHIER, and J. BENDER (2016). "Projective Fluids". *ACM SIGGRAPH Motion in Games*, pp. 1–6.

M. WEILER, D. KOSCHIER, M. BRAND, and J. BENDER (2018). "A Physically Consistent Implicit Viscosity Solver for SPH Fluids". *Computer Graphics Forum*.

M. WICKE, D. RITCHIE, B. M. KLINGNER, S. BURKE, J. R. SHEWCHUK, and J. F. O'BRIEN (July 2010). "Dynamic local remeshing for elastoplastic simulation". *ACM Transactions on Graphics* 29.4, 49:1–49:12.

C. WOJTAN and G. TURK (2008). "Fast Viscoelastic Behavior with Thin Features". *ACM Transactions on Graphics* 27.3, 47:1–47:8.

B. WU, Z. XU, and Z. LI (2008). "A note on imposing displacement boundary conditions in finite element analysis". *Communications in Numerical Methods in Engineering* 24.9, pp. 777–784.

J. WU, C. DICK, and R. WESTERMANN (2011). "Interactive High-Resolution Boundary Surfaces for Deformable Bodies with Changing Topology". *Proceedings of Virtual Reality Interactions & Physical Simulations (VRIPhys)*, pp. 29–38.

J. WU and L. KOBBELT (2003). "Piecewise Linear Approximation of Signed Distance Fields". *Vision, Modeling, and Visualization*, pp. 513–520.

X. WU, M. S. DOWNES, T. GOKTEKIN, and F. TENDICK (2001). "Adaptive Nonlinear Finite Elements for Deformable Body Simulation Using Dynamic Progressive Meshes". *Computer Graphics Forum* 20.3, pp. 349–358.

H. XU, Y. ZHAO, and J. BARBIČ (2014). "Implict Multibody Penalty-based Distributed Contact". *IEEE Transactions on Visualization & Computer Graphics* 20.9, pp. 1266–1279.

H. XU and J. BARBIČ (2014a). "Continuous Collision Detection Between Points and Signed Distance Fields". *Proceedings of Virtual Reality Interactions & Physical Simulations (VRIPhys)*.

H. XU and J. BARBIČ (2014b). "Signed Distance Fields for Polygon Soup Meshes". *Graphics Interface*, pp. 1–7.

L. YANG, S. LI, A. HAO, and H. QIN (Sept. 2012). "Realtime Two-Way Coupling of Meshless Fluids and Nonlinear FEM". *Computer Graphics Forum* 31.7, pp. 2037–2046.

Y.-H. YEUNG, J. CROUCH, and A. POTHEN (2016). "Interactively Cutting and Constraining Vertices in Meshes Using Augmented Matrices". *ACM Transactions on Graphics* 35.2, 18:1–18:17.

L. ZHANG, T. CUI, and H. LIU (2009). "A Set of Symmetric Quadrature Rules on Triangles and Tetrahedra". *Journal of Computational Mathematics* 27.1, pp. 89–96.

Y. ZHU and R. BRIDSON (2005). "Animating Sand as a Fluid". *ACM Transactions on Graphics* 24.3, pp. 965–972.

Y. ZHU, Y. WANG, J. HELLRUNG, A. CANTARERO, E. SIFAKIS, and J. M. TERAN (2012). "A second-order virtual node algorithm for nearly incompressible linear elasticity in irregular domains". *Journal of Computational Physics* 231.21, pp. 7092–7117.

G. ZI and T. BELYTSCHKO (2003). "New crack-tip elements for XFEM and applications to cohesive cracks". *International Journal for Numerical Methods in Engineering* 57.15, pp. 2221–2240.

O. ZIENKIEWICZ, R. TAYLOR, and J. ZHU (2013). *The Finite Element Method: its Basis and Fundamentals*. Elsevier Science, pp. 493–543.