ENGINEERING MATHEMATICS
AND COMPUTING LAB

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Simulation of Surgical Cutting in Soft Tissue using the Extended Finite Element Method (X-FEM)

Nicolai Schoch, Stefan Suwelack, Stefanie Speidel,
Rüdiger Dillmann, Vincent Heuveline

Preprint No. 2013-04

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)

www.emcl.iwr.uni-heidelberg.de

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)
ISSN 2191–0693
Preprint No. 2013-04

Affiliation of the Authors

Nicolai Schoch[a,1], Stefan Suwelack[b], Stefanie Speidel[b],
Rüdiger Dillmann[b], Vincent Heuveline[a]

[a] *Engineering Mathematics and Computing Lab (EMCL),*
Interdisciplinary Center for Scientific Computing (IWR),
Heidelberg University, Germany
[b] *Institute for Anthropomatics, Karlsruhe Institute of Technology (KIT), Germany*
[1] *Corresponding Author: Nicolai Schoch, nicolai.schoch@iwr.uni-heidelberg.de*

Impressum

Heidelberg University
Interdisciplinary Center for Scientific Computing (IWR)
Engineering Mathematics and Computing Lab (EMCL)

Speyerer Str. 6,
69115 Heidelberg
Germany

Simulation of Surgical Cutting in Soft Tissue using the Extended Finite Element Method (X-FEM)

Nicolai Schoch, Stefan Suwelack, Stefanie Speidel,
Rüdiger Dillmann, Vincent Heuveline

December 02, 2013

Abstract

Modeling and simulation of the behaviour of elastic bodies is an important tool for medical engineering. Most physics-based simulations use the finite element method (FEM) for simulating the behaviour of deformable soft tissue under the effect of external forces. However, especially the task of surgical cutting still remains an open challenge. Current methods which, e.g., require the adjustment of the mesh topology in order to align elements with the cut, still suffer from performance and stability issues. Opposed to this, the extended finite element method (X-FEM) provides a novel approach for modeling discontinuities without creating new mesh elements, and thus minimizes the impact on the performance.

We present the development of a 3D model for the simulation of cutting in soft tissue. The incorporation of a corotation-based formulation into a dynamic FEM simulation enables realistic material behaviour even under larger deformations, and guarantees stability as well as computationally efficient data structures and algorithms which allow for near-to-real-time frame rates.

On the basis of our elasticity simulation, we develop an extended model which allows for the simulation of surgical cutting in soft tissue using the X-FEM. As a distinguishing feature our implementation combines the corotational formulation and the implicit Newmark time integration method, which not only allows for the realistic simulation of large cutting-induced deformations but also results in very stable simulations of arbitrary cuts. For the X-FEM, too, the underlying data structures of our implementation yield a great potential for outsourcing computationally expensive calculations from the actual time-stepping scheme into a pre-computing part, and hence enhance the utility of this simulation for real-time applications.

The evaluation of our methods uses commercial FEM software for comparison and shows the convergence of our simulation results. The X-FEM even exhibits comparable accuracy in terms of DOFs with respect to perfectly remeshed standard FEM. Along with good performance and stability properties, this proves the applicability of the X-FEM-based cutting simulation, e.g., in surgery simulators.

1 Introduction and Motivation

1.1 Medical Context and Motivation

During the past decades, in the field of surgery assistence and simulation systems, a considerable number of instruments and techniques have been developed in an effort to improve processes in the medical,

clinical and surgical context. Especially, modeling and simulation of the behaviour of soft tissue in the human body opens up new opportunities. It not only allows surgeons to gain experience through virtual reality surgery training simulators, but also offers support during the actual surgical process by providing the surgeons via a virtual reality (VR) or augmented reality (AR) with 3D models of the body parts which are operated on.

A wide range of VR and AR applications in the medical context are therefore currently improved and extended by a substantial amount of additional features, which, e.g., allow the simulation of deformations of soft tissue under the effect of external forces during cutting processes. However, particularly the simulation of cutting still remains an open challenge, since most current methods suffer from both performance and stability issues, impeding their application in the medical real-time context.

In this work, we present a 3D model and simulation of the behaviour of soft tissue subject to external forces, which is based on the corotational formulation of the finite element method (FEM). It hence not only allows for a realistic material behaviour even when given large deformations, but also turns out to be computationally efficient anad stable. The simulation is extended by means of the extended finite element method (X-FEM) in order to allow for discontinuities, thus representing cracks and cuts. The implementation applies the implicit Newmark time integration scheme for time discretization and operates on a novel data structure, which yields very good convergence properties and generally extremely stable results in near-to-real-time frame rates.

1.2 Related Work and State-of-the-art of Science and Technology

There is a wide range of simulation applications in the field of medical engineering, that benefit from physically-based modelling, most of which use the FEM for numerically solving the given elasticity problem. An essential feature of a present-day medical simulation system is the capability to simulate surgical cuts. Yet, this still remains a challenging task, since the overall goal is to allow for arbitrary object dissection without suffering from restrictions, such as performance and stability issues.

Most of the methods for simulating surgical cutting published so far require the elements to align with the cut. In order to achieve this alignment, there are several options. Han-Wen Nienhuys and his colleagues suggest approaches based on constraining the cut to the borders of existing elements, or vice versa snapping the elements' borders to the cut, compare [17]. These approaches do not create new elements and therefore keep the computation simple, yet they mostly do not satisfy accuracy requirements due to projection errors. Other methods suggest the elements which are cut to be subdivided, compare e.g. Mazura [14]. The approach of Daniel Bielser and his colleagues subdivides elements according to a set of predefined templates, [5]. The main drawback of these methods is the creation of small ill-shaped elements (slivers) that cause numerical instability in the simulation. Moreover, since disproportionally many new (sub)element nodes have to be introduced, the system matrices must be extended, too, which causes an explosion of the amount of work needed for the calculations. Stephane Cotin and his colleagues propose an approach, which consists of removing the elements touched by the cutting surface or by the medical cutting tools, [8]. This approach does not harm the simulation stability, however, it violates the mass conservation law, since the open space is not filled anymore, also leaving visually unpleasant artifacts. Meshless methods, as presented by Belytschko, [3], allow the simulation of deformable objects including topological changes such as arising from cutting. However, they are computationally significantly more expensive than FEM. Summarizing, there are many approaches to model discontinuities representing surgical cuts in medical simulations. However, the underlying problems and drawbacks have not been solved satisfyingly yet.

Opposed to this, the X-FEM, proposed by Ted Belytschko and his colleagues, uses element enrichment to effectively model discontinuity regions within an FEM mesh, [4]. The local enrichment, which is based on the partition of unity concept, is introduced only in subregions with discontinuities. In combination with an appropriate mass-lumping technique, the X-FEM provides a stable simulation regardless of the cut location. Belytschko first developed the X-FEM for 2D linear elastic fracture mechanics in civil engineering, considering a single crack. Subsequently, the method has been extended to many applications, such as dynamic crack growth, arbitrarily branched and intersecting cracks or holes, and even weak discontinuities, in which the displacement is continuous but the displacement gradient contains a discontinuity, such as when regarding material interfaces in solids and fluids. Its general applicability, along with the fact that it does not require remeshing, let the X-FEM become a highly appreciated method for

the purpose of cutting simulations.

Finally, a research team around Lara Vigneron presents an X-FEM-based approach for modeling the deformation of organs following surgical cuts, retractions, and resections where arbitrarily-shaped tissue discontinuities are handled via the X-FEM without any remeshing, [18].

2 Basics of Elasticity Theory and Soft Tissue Simulation

In a brief overview of elasticity theory, we will provide the reader with the necessary biophysical principles and general modeling techniques used for the simulation of the behaviour of soft bodies under the effect of external load as far as it is relevant for this work, and show a specific mathematical formulation for the so-called elasticity problem, which later allows for calculating the deformation of the elastic body using finite element methods (FEM). For further information we refer to [6] and [7].

2.1 Basics of Elasticity Theory

In elasticity theory (or generally in continuum mechanics) a body is seen as a continuum, i.e., a set of material points in space that are linked to each other, and its configuration is observed under the impact of external forces. In particular, we measure strains and stresses caused by deformations. The essential components of the elasticity theory are the kinematics, the equilibrium equations and the constitutive (material) laws.

In an abstract view of elasticity theory, we consider a body's stress-free reference configuration Ω and a deformation $\boldsymbol{\varphi}$ which transforms it into its deformed configuration $\tilde{\Omega}$ as follows

$$\boldsymbol{\varphi}: \begin{cases} \Omega \to \tilde{\Omega} \subset \mathbb{R}^3 \\ \boldsymbol{x} \mapsto \boldsymbol{\varphi}(\boldsymbol{x}) = \tilde{\boldsymbol{x}} \end{cases} . \tag{1}$$

The behaviour of a body can be described by the relationship between the four physical quantities displacement, strain, stress and force. In kinematics, the *displacement* $\boldsymbol{u}$ of a body's particle at point $\boldsymbol{x}$, which hence corresponds to $\boldsymbol{u} = \boldsymbol{\varphi} - \mathrm{id}$, is linked to the internal deformation measure *strain* $\boldsymbol{\varepsilon}$, a second-order tensor, by kinematic compatibility conditions

$$\boldsymbol{\varepsilon}(\nabla\boldsymbol{\varphi}) = \frac{1}{2}\left((\nabla\boldsymbol{\varphi})^T(\nabla\boldsymbol{\varphi}) - \boldsymbol{I}\right) , \quad \text{or} \quad \boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}(D\boldsymbol{u} + D\boldsymbol{u}^T) + \frac{1}{2}\sum D\boldsymbol{u}D\boldsymbol{u}^T , \tag{2}$$

where $\nabla\boldsymbol{\varphi}$ is the *deformation gradient*. When neglecting quadratic terms, this yields the following approximation, known as the *linearized Cauchy strain tensor*

$$\boldsymbol{\varepsilon}^{\mathrm{lin}}(\nabla\boldsymbol{\varphi}) = \frac{1}{2}\left((\nabla\boldsymbol{\varphi})^T + \nabla\boldsymbol{\varphi}\right) - \boldsymbol{I} , \quad \text{or} \quad \boldsymbol{\varepsilon}^{\mathrm{lin}}(\boldsymbol{u}) = \frac{1}{2}(D\boldsymbol{u} + D\boldsymbol{u}^T) . \tag{3}$$

The internal force measure *stress* $\boldsymbol{\sigma}$ is linked to the *strain* $\boldsymbol{\varepsilon}$ by constitutive (material) laws via a response function C as follows

$$\boldsymbol{\sigma}(\boldsymbol{x}) = C(\boldsymbol{x}, \boldsymbol{\varepsilon}(\boldsymbol{x})), \quad \boldsymbol{x} \in \Omega. \tag{4}$$

In this work, we restrict our considerations to *linear isotropic materials*, which are characterized by a linearized relation between stress and strain, and by the feature that the properties of all material points in all space directions are equal. Moreover, we assume the material to be *homogeneous*, which means that the material properties do not explicitly depend on the concerned point $\boldsymbol{x}$ of the object. By exploiting symmetries and geometric relations, these simplifications, i.e., the use of linear isotropic materials, allow to reduce the number of coefficients of the forth-order tensor C from originally 81 entries to only two independent entries in its matrix representation. The restrictions of this so-called *linear elasticity theory* are not only due to reasons of simplicity and suitability for real-time applications, but also recommendable from the modeling point of view, as the isotropic case preponderates for small deformations in reality, where non-linear phenomena usually do not yet occur. However, we will also show the limits of these assumptions later in this work and present a solution based on co-rotation.

In the linear elasticity theory, the relation of stress and strain, the so-called *linear material law of Hooke*, reads as

$$\boldsymbol{\sigma}^{\text{lin}} = \lambda \operatorname{tr}(\boldsymbol{\varepsilon}^{\text{lin}})\boldsymbol{I} + 2\mu\boldsymbol{\varepsilon}^{\text{lin}}, \tag{5}$$

where λ and μ are the *Lame constants*, which account for characteristic material properties and allow to influence the behaviour of the material which is modeled. Alternatively, one could employ the engineering constants E (the elasticity or Young modulus) and ν (the transverse contraction or Poisson ratio).

Using *Voigt notation*, equation (5) can be written in a matrix form, where the second-order tensors $\boldsymbol{\sigma}^{\text{lin}}$ and $\boldsymbol{\varepsilon}^{\text{lin}}$ are rearranged as 6-component vectors and the forth-order tensor response function C is represented by the so-called *material matrix* $\mathbb{C}$ as follows

$$\boldsymbol{\sigma}^{\text{lin}} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{xz} \\ \tau_{yz} \end{bmatrix} = \underbrace{\begin{bmatrix} 2\mu+\lambda & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & 2\mu+\lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & 2\mu+\lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix}}_{=\mathbb{C}} \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \end{bmatrix} = \mathbb{C}\,\boldsymbol{\varepsilon}^{\text{lin}}, \tag{6}$$

where

$$\varepsilon_x = \varepsilon_{xx}\,,\quad \varepsilon_y = \varepsilon_{yy}\,,\quad \varepsilon_z = \varepsilon_{zz}\,,\quad \gamma_{xy} = \varepsilon_{xy} + \varepsilon_{yx}\,,\quad \gamma_{xz} = \varepsilon_{xz} + \varepsilon_{zx}\,,\quad \gamma_{zy} = \varepsilon_{zy} + \varepsilon_{yz}\,.$$

Finally, the stress $\boldsymbol{\sigma}$ is linked to the external load $\boldsymbol{b}$ by the equilibrium equations. Forces and displacements can be observed and measured as external factors which can be prescribed by defining boundary conditions (BC), corresponding, e.g., to the fixing of a part of an object (displacement or Dirichlet BC) or the application of an interaction force (force or Neumann BC). The acting forces can be completely traced back to forces which act on surfaces (*surface forces*, e.g. the load on the abdominal organs which is produced by the lungs when expanding during the process of breathing) and forces which act on volumes (*volume forces*, e.g. gravity).

Strain and stress are internal mathematical tools to measure the effect of deformations, i.e., displacements and forces, respectively. The relationship between stress and strain is what determines the actual physical behaviour of the body as a continuum. This work is based on the linearized consideration; In the following, we will, however, not differentiate between the notations of the normal and the linearized versions of strain and stress anymore.

2.2 Differential and Variational Formulation of the Elasticity Problem

Based on the above introduced assumptions, conditions and dependencies, we can now define the so-called *boundary value problem (BVP)* of elasticity theory according to the *Theorem of Cauchy*:

Find the displacement $\boldsymbol{u} = \boldsymbol{\varphi} - \text{id} \in \mathbb{C}^2(\Omega) \cap \mathbb{C}^1(\overline{\Omega})$ such that

$$-\operatorname{div}(\boldsymbol{\sigma}(\boldsymbol{x})) = \boldsymbol{b}(\boldsymbol{x})\,, \quad \boldsymbol{x} \in \Omega\,, \tag{7}$$

$$\boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{u_0}(\boldsymbol{x})\,, \quad \boldsymbol{x} \in \Gamma_D\,, \tag{8}$$

$$\boldsymbol{\sigma}(\boldsymbol{x}) \cdot \boldsymbol{n}(\boldsymbol{x}) = \boldsymbol{s}(\boldsymbol{x})\,, \quad \boldsymbol{x} \in \Gamma_N^{\neq 0}\,, \tag{9}$$

where $\boldsymbol{b} : \Omega \to \mathbb{R}^3$ denotes the volume force acting on whole Ω, $\boldsymbol{s} : \Gamma_N^{\neq 0} \to \mathbb{R}^3$ stands for the surface force in direction of normal vector $\boldsymbol{n}$ on the boundary part $\Gamma_N^{\neq 0}$ (*Neumann BC*), and finally Γ_D is the boundary part where the displacement $\boldsymbol{u_0} : \Gamma_D \to \mathbb{R}^3$ is given (*Dirichlet BC*). We define $\Gamma_N = \partial\Omega \backslash \Gamma_D$, so we need $\boldsymbol{s}(\boldsymbol{x}) = 0\,,\ \boldsymbol{x} \in \partial\Omega \backslash (\Gamma_D \cup \Gamma_N^{\neq 0})$, such that $\Gamma_D \cap \Gamma_N = \emptyset$ holds.

For a simplified formulation we define the following spaces $\boldsymbol{V}_B(\Omega, \boldsymbol{f}) = \{\boldsymbol{u} | -\operatorname{div}(\boldsymbol{\sigma})_{|\Omega} = \boldsymbol{f}\}$, $\boldsymbol{V}_N(\Gamma_N, \boldsymbol{f}) = \{\boldsymbol{u} | (\boldsymbol{\sigma} \cdot \boldsymbol{n})_{|\Gamma_N} = \boldsymbol{f}\}$, and $\boldsymbol{V}_D(\Gamma_D, \boldsymbol{f}) = \{\boldsymbol{u} | \boldsymbol{u}_{|\Gamma_D} = \boldsymbol{f}\}$ for an arbitrary function $\boldsymbol{f}$, as well as $\boldsymbol{V} := \mathbb{C}^2(\Omega) \cap \mathbb{C}^1(\overline{\Omega}) \cap \boldsymbol{V}_B(\Omega, \boldsymbol{b}) \cap \boldsymbol{V}_N(\Gamma_N, \boldsymbol{s}) \cap \boldsymbol{V}_D(\Gamma_D, \boldsymbol{u}_0)$, such that the above formulation of the elasticity problem now presents as:

Find the displacement $\boldsymbol{u} = \boldsymbol{\varphi} - \text{id} \in \boldsymbol{V}$.

By means of the definition of a suitable test function space according to $\boldsymbol{V}$, multiplication with test functions, integration over Ω, and the integration theorem of Gauss, the differential formulation of the

BVP can then be transformed into the *weak or variational formulation*, which considers a system's total energy Π, and finally claims its stationarity with respect to the deformation $\boldsymbol{\varphi}$ or the displacement $\boldsymbol{u}$ wanted.

The *system's total energy* Π of the deformed configuration can be composed of the total internal energy W and the sum of all external (volume and surface) forces acting on the body. Using the Frobenius Product[1] we find

$$\Pi^{\mathrm{stat}}(\boldsymbol{u}) = \underbrace{\int_\Omega \boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\sigma}(\boldsymbol{u})\, d\Omega}_{=W^{\mathrm{stat}}(\boldsymbol{u})} - \underbrace{\int_\Omega \boldsymbol{u}^T \boldsymbol{b}\, d\Omega}_{\text{volume force}} - \underbrace{\int_{\Gamma_1} \boldsymbol{u}^T \boldsymbol{s}\, d\Gamma}_{\text{surface force}}, \tag{10}$$

in the static case, and – additionally accounting for viscous stress and inertia contributions –

$$\begin{aligned} \Pi^{\mathrm{dyn}}(\boldsymbol{u}) \quad = \quad & \underbrace{\underbrace{\int_\Omega \boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\sigma}(\boldsymbol{u})\, d\Omega}_{\text{stiffness}} + \underbrace{\int_\Omega \dot{\boldsymbol{\varepsilon}}(\dot{\boldsymbol{u}}) : \boldsymbol{\sigma}_{\boldsymbol{v}}(\dot{\boldsymbol{u}})\, d\Omega}_{\text{viscosity}} + \underbrace{\int_\Omega \frac{1}{2}\rho\, |\, \ddot{\boldsymbol{u}}^2\, |\, d\Omega}_{\text{inertia}}}_{=W^{\mathrm{dyn}}(\boldsymbol{u})} \\ & - \underbrace{\int_\Omega \boldsymbol{u}^T \boldsymbol{b}\, d\Omega}_{\text{volume force}} - \underbrace{\int_{\Gamma_1} \boldsymbol{u}^T \boldsymbol{s}\, d\Gamma}_{\text{surface force}}, \end{aligned} \tag{11}$$

in the dynamic case, with the damping tensor $\boldsymbol{\sigma}_{\boldsymbol{v}}$ and the mass density ρ.

Following physical laws, a system always aims to reach a state of minimum energy:

$$\Pi(\boldsymbol{u}) \stackrel{!}{=} \min. \tag{12}$$

Subsequently carrying out a variation and solving $\delta\Pi = 0$ according to *Hamilton's Principle* yields the so-called *Equation of Virtual Work*:

$$\begin{aligned} \delta\Pi^{\mathrm{dyn}} \quad = \quad & \int_\Omega \delta\boldsymbol{\varepsilon}^{\mathrm{elast}} : \boldsymbol{\sigma}\, d\Omega + \int_\Omega \delta\boldsymbol{\varepsilon}^{\mathrm{visc}} : \boldsymbol{\sigma}^{\mathrm{visc}}\, d\Omega + \int_\Omega \delta\boldsymbol{u}^T\, \rho\, \ddot{\boldsymbol{u}}\, d\Omega \\ & - \int_\Omega \delta\boldsymbol{u}^T \boldsymbol{b}\, d\Omega - \int_\Gamma \delta\boldsymbol{u}^T \boldsymbol{s}\, d\Gamma \qquad\qquad = 0\,. \end{aligned} \tag{13}$$

This variational formulation of the elasticity problem finally serves for solving by means of numerical methods, such as the *Finite Element Method* (FEM). More information on methods for the solution of mathematical models of systems in continuum mechanics can be found, e.g., in [1].

3 Numerical Solution of the Elasticity Problem using FEM

This section presents a strongly simplified view of the FEM, and sketches the corresponding quantities and data structures applied in the implementation of the elasticity simulation in a way which is sufficient to understand this work. More details on the FEM can be found in standard FEM books, e.g., in [1], [21].

3.1 FE discretization based on linear and quadratic elements

In order to numerically solve the variational formulation of the elasticity problem in $\boldsymbol{V}$, a subspace with finite dimensions has to be defined. Using a *Ritz-Galerkin method*, the discretization arises from a continuous partitioning of the domain into finitely many elements[2] with locally defined *shape functions*[3] $\Phi_i : \mathbb{R}^3 \to \mathbb{R}^3$ that interpolate the *vertices* $\boldsymbol{P}$, $\tilde{\boldsymbol{P}} \in \mathbb{R}^{n\times 3}$ (before and after the deformation) and the

[1] The Frobenius inner product is defined as: $\boldsymbol{A} : \boldsymbol{B} = \mathrm{tr}(\boldsymbol{A}^T\boldsymbol{B}) = \sum_{i,j} A_{ij}B_{ij}$.

[2] In this work: tetrahedra.

[3] For standard definitions of FE shape functions, see, e.g., [1].

corresponding *nodal displacements* $\boldsymbol{\mathcal{U}} = \tilde{\boldsymbol{P}} - \boldsymbol{P} \in \mathbb{R}^{n\times 3}$, where n is the number of element nodes. We find the displacement $\boldsymbol{u}(\boldsymbol{x})$ of an arbitrary point $\boldsymbol{x}$ given as

$$\boldsymbol{u}(\boldsymbol{x}) = \sum_{i=1}^{n} \boldsymbol{\mathcal{U}}_i \Phi_i(\boldsymbol{x})\,, \tag{14}$$

where the Φ_i satisfy the *partition of unity* $\sum_{i=1}^{n} \Phi_i = 1$ and the *Kronecker delta property* $\Phi_i(\boldsymbol{P}_j) = \delta_{ij}$. We state

$$\nabla \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{\mathcal{U}} \nabla \Phi(\boldsymbol{x})\,, \quad \nabla \boldsymbol{\varphi}(\boldsymbol{x}) = \nabla \boldsymbol{u}(\boldsymbol{x}) + \boldsymbol{I} = \boldsymbol{\mathcal{U}} \nabla \Phi(\boldsymbol{x}) + \boldsymbol{I}\,. \tag{15}$$

For our implementation we provide so-called *isoparametric shape functions* Φ_i, and – even more common – the corresponding *isoparametric finite elements*, which means that both, the transformation function for the mapping from a real global element to a local reference element – as will be explained below –, and the shape functions for the interpolation of the deformation are of the same degree. Using these isoparametric elements for the FE formulation allows the achievement of a relationship between the element displacement at any arbitrary point and the element nodal point displacements directly through the use of the shape functions for interpolation.

On the element level, isoparametric elements were implemented by the example of linear and quadratic tetrahedra, hence $\boldsymbol{P} \in \mathbb{R}^{4\times 3}$ or $\boldsymbol{P} \in \mathbb{R}^{10\times 3}$. Linear basis functions are obviously easier to handle with respect to computing and implementation, however, in engineering applications, the use of linear basis functions is usually avoided, firstly, since they only achieve weak convergence, caused by $\nabla\boldsymbol{\varphi}$ being constant and consequently – as will be explained below – also the stiffness matrix being constant on the whole element, and secondly, since in case of almost incompressible materials – which we deal with during our simulations – linear elements suffer from so-called *numerical locking effects*.

Using quadratic shape functions, we reduce approximation errors inherent to methods based on linear finite elements. In comparison to linear isoparametric elements they not only offer better numerical accuracy by means of quadratic shape functions but also superior geometric approximation by means of their quadratic element geometry, even with a much smaller number of elements, compare [16].

3.2 Discretization of Elastic Force Density, Inertia and External Force Densities

Based on the introduced notation and the hence arising data structures to be used in the implementation, we can proceed with the discretization of elastic forces, inertia and the external load as given in the variational formulation of the elasticity problem (13). We consider (13) elementwise.

Starting with the stiffness part, the integrand is transformed as follows

$$\delta\boldsymbol{\varepsilon} : \boldsymbol{\sigma} \overset{\text{symmetry of } \boldsymbol{\sigma}}{=} (\nabla\boldsymbol{\varphi}^T \nabla \delta\boldsymbol{u}) : \boldsymbol{\sigma} \overset{\nabla\boldsymbol{u}=\boldsymbol{\mathcal{U}}^T\nabla\Phi}{=} (\delta\boldsymbol{\mathcal{U}}^T \nabla\Phi) : (\nabla\boldsymbol{\varphi}\,\boldsymbol{\sigma}) \overset{\boldsymbol{\mathcal{F}}:=\nabla\Phi\,\boldsymbol{\sigma}\,\nabla\boldsymbol{\varphi}^T}{=} \delta\boldsymbol{\mathcal{U}} : \boldsymbol{\mathcal{F}}\,, \tag{16}$$

with the *element force matrix* $\boldsymbol{\mathcal{F}} \in \mathbb{R}^{n\times 3}$, which holds the n nodal force densities at the element vertices of the isoparametric tetrahedron. It can be shown that the ordering of the factors is crucial for an efficient computation, and that by choosing

$$\boldsymbol{\mathcal{F}} = \left[\sum_{k=1}^{3} \nabla\boldsymbol{\varphi}_{jk} \sum_{l=1}^{3} \nabla\Phi_{il}\boldsymbol{\sigma}_{lk} \right]_{i=1\ldots n,\, j=1\ldots 3}\,, \tag{17}$$

the smallest number of floating point operations is obtained, compare [15]. An analogue treatment of the inertia term results in

$$\delta\boldsymbol{u}^T \rho\, \ddot{\boldsymbol{u}} = \Phi^T \delta\boldsymbol{\mathcal{U}}\, \rho \ddot{\boldsymbol{\mathcal{U}}}^T \Phi \overset{\boldsymbol{\mathcal{M}}:=\rho\,\Phi\Phi^T}{=} \delta\boldsymbol{\mathcal{U}} : (\boldsymbol{\mathcal{M}}\ddot{\boldsymbol{\mathcal{U}}})\,, \tag{18}$$

with the symmetric element mass matrix $\boldsymbol{\mathcal{M}} \in \mathbb{R}^{n\times n}$. Assuming the external body and surface forces to be *dead loads*, i.e., independent of the deformation φ, we can derive

$$\delta\boldsymbol{u}^T\,\boldsymbol{b} \overset{\boldsymbol{\mathcal{B}}:=\Phi\,\boldsymbol{b}^T}{=} \delta\boldsymbol{\mathcal{U}} : \boldsymbol{\mathcal{B}} \quad \text{and} \quad \delta\boldsymbol{u}^T\,\boldsymbol{s} \overset{\boldsymbol{\mathcal{S}}:=\Phi\,\boldsymbol{s}^T}{=} \delta\boldsymbol{\mathcal{U}} : \boldsymbol{\mathcal{S}}\,, \tag{19}$$

where the body and surface force matrices $\boldsymbol{\mathcal{B}}$, $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{n\times 3}$. Finally, the damping term is transformed using the approach of Rayleigh, which suggests the damping matrix $\boldsymbol{\mathcal{D}}$ to be a linear combination of the above calculated elastic force and mass matrices $\boldsymbol{\mathcal{F}}$ and $\boldsymbol{\mathcal{M}}$ as follows

$$\boldsymbol{\mathcal{D}} := \alpha\boldsymbol{\mathcal{M}} + \beta\boldsymbol{\mathcal{F}}\,, \tag{20}$$

where the coefficients α and β (*Rayleigh coefficients*) can be determined experimentally. This simplification also allows for a more compute and time efficient implementation.

3.3 The Virtual Work ODE

Summarizing the above results yields

$$\delta\Pi^{\text{dyn}} = \underbrace{\int_V \delta\boldsymbol{\mathcal{U}} : \boldsymbol{\mathcal{F}}\, dV}_{\text{stiffness term}} + \underbrace{\int_V \delta\boldsymbol{\mathcal{U}} : \boldsymbol{\mathcal{D}}\, dV}_{\text{damping term}} + \underbrace{\int_V \delta\boldsymbol{\mathcal{U}} : \boldsymbol{\mathcal{M}}\ddot{\boldsymbol{\mathcal{U}}}\, dV}_{\text{mass term}} - \underbrace{\int_V \delta\boldsymbol{\mathcal{U}} : \boldsymbol{\mathcal{B}}\, dV - \int_A \delta\boldsymbol{\mathcal{U}} : \boldsymbol{\mathcal{S}}\, dA}_{\text{external load term}} = 0\,, \tag{21}$$

where the variation does not depend on the integrals, and which must hold for any $\delta\boldsymbol{\mathcal{U}}$, such that we can equivalently rearrange the equation into the second order ODE

$$\begin{aligned} 0 &= \int_V \boldsymbol{\mathcal{F}}\, dV + \int_V \boldsymbol{\mathcal{D}}\, dV + \int_V \boldsymbol{\mathcal{M}}\ddot{\boldsymbol{\mathcal{U}}}\, dV - \int_V \boldsymbol{\mathcal{B}}\, dV - \int_A \boldsymbol{\mathcal{S}}\, dA \\ &=: \boldsymbol{F}(\boldsymbol{\mathcal{U}}) + \boldsymbol{D}(\dot{\boldsymbol{\mathcal{U}}}) + \boldsymbol{M}\ddot{\boldsymbol{\mathcal{U}}} - \boldsymbol{B} - \boldsymbol{S}\,. \end{aligned} \tag{22}$$

In the FE theory, due to the elements' compact supports and the C^0 continuity of the tetrahedralization, the volume integrals can be evaluated element-wise, and the resulting element matrices can be merged into a single global matrix with three coordinates per mesh vertex by means of lexicographical rearrangements. In other words, for the entries $\boldsymbol{F}_{lj}$ of the global stiffness matrix $\boldsymbol{F}$ we find

$$\boldsymbol{F}_{lj} = \sum_k \int_{V_k} \boldsymbol{\mathcal{F}}_{ij}\, dV_k\,, \quad \text{where } l = 1\dots N,\ i = 1\dots n,\ j = 1\dots 3, \tag{23}$$

i.e., the sum of nodal elastic forces $\boldsymbol{\mathcal{F}}_{ij}$ from all the elements k containing the vertex l in any i-th row. Again, in our notation n denotes the number of nodes per element, N denotes the object's total number of nodes, and the factor 3 accounts for the 3D space.

3.4 Volume Integration

In order to achieve near-to-real-time simulation through a fast set-up of the system matrices and hence the above equations, it has to be taken care of an efficient numerical volume integration of the element matrices first. Today's standard FEM codes generally trace the computation of an integral over an arbitrary element in a global context back to the calculation of this integral over a (local, mostly simple) reference element, which in case of linear elements is done by means of an affine transformation $\hat{\boldsymbol{\varphi}}$. Subsequently, the cubature is based on the standard Gauss-Legendre integration rule. This allows for the best possible precision with respect to a given number q of integration points $\boldsymbol{\zeta}_i$, $i = 1...q$ (compare [1]), where in this work we use $q = 4$ for mass matrices, and $q = 1$ for stiffness matrices:

$$\int_V f(\boldsymbol{x})\, dV \overset{\text{transformation \& approximation}}{\approx} \sum_{i=1}^{q} \omega_i\, f(\hat{\varphi}(\boldsymbol{\mathcal{T}}, \boldsymbol{\zeta}_i))\, \det(\nabla\hat{\boldsymbol{\varphi}}(\boldsymbol{\mathcal{T}}, \boldsymbol{\zeta}_i))\,. \tag{24}$$

It is recommended to consider the precomputation of the transformation determinants $\det(\nabla\hat{\boldsymbol{\varphi}}(\boldsymbol{\mathcal{T}}, \boldsymbol{\zeta}_i))$, shape functions $\Phi(\hat{\boldsymbol{\varphi}}(\boldsymbol{\mathcal{T}}, \boldsymbol{\zeta}_i))$ and their derivatives $\nabla\Phi(\hat{\boldsymbol{\varphi}}(\boldsymbol{\mathcal{T}}, \boldsymbol{\zeta}_i))$ with respect to the reference tetrahedron for each cubature point in a preprocessing step once and storing them for later use, which allows for efficiently updating the matrices. Concerning the computation of the element stiffness matrices, it is important to mention, that even when using linear shape functions and hence all terms in $\boldsymbol{\mathcal{F}}$ are constant on the whole tetrahedron, $\text{vol}(\boldsymbol{\mathcal{T}})$ and $\nabla\Phi$ are still worthwhile for preprocessing.

Addressing the computation of the element mass matrices, we consider the subsequent time integration, where the inverse of the mass matrix is needed. Knowing this, it would be convenient to have a diagonal – and hence invertible – mass matrix, allowing for stable simulations. Yet, a diagonal mass matrix means that the mass is concentrated at the mesh vertices, which contradicts the idea of continuum mechanics and finite elements, where the mass distribution is continuous. Therefore, for efficiency and applicability reasons, the use of orthogonal basis functions can be recommended, since the local mass matrices then directly result in diagonal matrices, which implies that this also holds for the global mass matrix. In contrast, when using the above introduced nodal shape functions, the diagonalisation has to be conducted seperately by means of so-called *mass lumping methods*.

3.5 Time Integration

In an interactive simulation when we are interested in the behaviour of a deformed body over time, the simulation has to be based on the dynamical description as given in the second order ODE (22). This hence requires time integration in addition to space discretization.

After summing up the element force matrices $\boldsymbol{B}$ and $\boldsymbol{S}$ to $\boldsymbol{F}_{ext}$, and lexicographically re-sorting the element matrices $\boldsymbol{F}_{ext}$, $\ddot{\boldsymbol{\mathcal{U}}}$, $\dot{\boldsymbol{\mathcal{U}}}$ and $\boldsymbol{\mathcal{U}}$ for all elements into vectors $\boldsymbol{f}_{ext}$, $\ddot{\boldsymbol{u}}$, $\dot{\boldsymbol{u}}$ and $\boldsymbol{u}$ (each $\in \mathbb{R}^{3N}$), we can rewrite (22) using matrix-vector-notation, which results in

$$\mathbb{M}\ddot{\boldsymbol{u}} + \mathbb{D}\dot{\boldsymbol{u}} + \mathbb{K}\boldsymbol{u} = \boldsymbol{f}_{ext}\,, \tag{25}$$

where $\mathbb{M}$ is the global mass matrix, $\mathbb{D}$ is the damping matrix, $\mathbb{K}$ is the global stiffness matrix, $\mathbb{M}$, $\mathbb{D}$, $\mathbb{K} \in \mathbb{R}^{3\cdot N\times 3\cdot N}$, $\boldsymbol{u}$ is the vector of nodal displacements, and $\boldsymbol{f}_{\text{ext}} \in \mathbb{R}^{3N}$ is the external load vector, including both body and surface loads. The global matrices are added up from the element matrices ${}_{elem}\mathbb{M}$, ${}_{elem}\mathbb{D}$ and ${}_{elem}\mathbb{K}$ (each $\in \mathbb{R}^{3\cdot n\times 3\cdot n}$) component-by-component at the lexicographically corresponding entry. In detail, for the $[3\times 3]$ components of the element matrices, we have

$${}_{elem}\mathbb{M}_{ij} := \rho \int_V \Phi_i \Phi_j \, dV \tag{26}$$

and retrieve

$${}_{elem}\mathbb{K}_{ij} := \left(\nabla_{\boldsymbol{\mathcal{U}}}\boldsymbol{F}\right)_{ij}\,, \tag{27}$$

where according to (22) $\boldsymbol{F}$ corresponds to the entries of the nodal elastic forces with respect to their global lexicographical order.

Rearranging equation (25) in terms of the acceleration yields

$$\boldsymbol{a} = \ddot{\boldsymbol{u}} = \mathbb{M}^{-1}\left(\boldsymbol{f}_{\text{ext}} - \mathbb{D}\dot{\boldsymbol{u}} - \mathbb{K}\boldsymbol{u}\right)\,, \tag{28}$$

which can be integrated in time twice in order to obtain the nodal displacements $\boldsymbol{u}$.

The choice of the numerical time integration method has to meet the specific requirements of an interactive medical simulation application. The most essential of these requirements are the simulation's accuracy, stability and real-time capability, the latter of which is dependent on the amount of work of the simulation steps. So-called direct integration methods, which are subclassified into explicit and implicit time integration methods, are considered most efficient with respect to these requirements.

Explicit methods are generally easy to implement, as in each time step only the information from previous steps is used. However, a major disadvantage of these techniques is their conditional stability, meaning that the time steps Δt_{step} are limited by a critical simulation time step Δt_{crit}, above which they become unstable. However, for a stable computation of the new values of acceleration and displacement, Δt_{step} must be larger than the time needed to compute these and transfer them $\Delta t_{\text{compute}}$. In other words, $\Delta t_{\text{compute}} < \Delta t_{\text{step}}$ and $\Delta t_{\text{step}} < \Delta t_{\text{crit}}$ must hold for a real time simulation. Hence, if, for the sake of stability, Δt becomes too small, even a method with a simple single step can become too computationally expensive for real time applications.

Therefore, in biomechanical simulations, we rather employ implicit methods, which can offer unconditional stability such that the time step Δt_{step} can be chosen a lot larger than in explicit methods. However, in comparison to explicit methods, they require a by far larger amount of work for their computations due to the fact that in every time step a system of equations has to be solved for the evaluation of partial

derivatives of the deformation forces and thus for the assembly of the tangent stiffness matrix. However, in case the stiffness matrix is constant, which it is in linear elasticity, the derivatives do not change in time either, which means that the time-consuming part of their calculation is not required at run-time. It is performed once only before the actual time stepping at the beginning of the simulation.

In our simulation, we apply the so-called *Newmark implicit integration method*, for the basic integration scheme of which we refer to [1]. We remark, that we make efficient use of precomputable entities such as mentioned above in a preprocessing part in order to minimize the computational overhead for the iteration steps whenever a new stiffness matrix is needed, and that visualization is generally outsourced into a seperate postprocessing part.

3.6 Calculation and Assembly of the Elasticity Stiffness Matrix

In order to allow for optimal performance and stability, we consider the process of assembling and calculating the system matrices, in particular the stiffness matrix $\mathbb{K}$ as given in equation (25). Hence, from equation (27) we find the $[3n \times 3n]$ element stiffness matrix

$$_{elem}\mathbb{K} = \nabla_{\boldsymbol{\mathcal{U}}} \boldsymbol{F} = \frac{\partial \boldsymbol{\mathcal{F}}}{\partial \boldsymbol{\mathcal{U}}} = \frac{\partial \boldsymbol{\mathcal{F}}}{\partial \tilde{\boldsymbol{\mathcal{P}}}}\,, \tag{29}$$

which can be derived by means of inserting the definition of the displacement $\boldsymbol{\mathcal{U}}(\boldsymbol{\mathcal{P}}) = (\boldsymbol{\varphi}-\mathrm{id})(\boldsymbol{\mathcal{P}}) = \tilde{\boldsymbol{\mathcal{P}}}-\boldsymbol{\mathcal{P}}$, or $\boldsymbol{\mathcal{U}}(\tilde{\boldsymbol{\mathcal{P}}}) = \tilde{\boldsymbol{\mathcal{P}}} - \boldsymbol{\mathcal{P}}$ (where $\boldsymbol{\mathcal{P}}$ is a constant), respectively.

As stated above, we assume that only small deformations occur, and thus employ the linear elasticity theory, exploiting the modeling and computational advantages arising through the afore-mentioned set of simplifications. Thus, we let $\boldsymbol{\mathcal{F}}^{\mathrm{lin}} = \nabla\Phi\boldsymbol{\sigma}^{\mathrm{lin}}$, which was derived in the same way as $\boldsymbol{\mathcal{F}} = \nabla\Phi\,\boldsymbol{\sigma}\,\nabla\boldsymbol{\varphi}^T$, however, using linear relations for $\boldsymbol{\varepsilon}^{\mathrm{lin}}$ and $\boldsymbol{\sigma}^{\mathrm{lin}}$. The components of the linearized element stiffness matrix $_{elem}\mathbb{K}^{\mathrm{lin}}$, i.e., the linearized version of the force derivatives $\frac{\partial \boldsymbol{\mathcal{F}}^{\mathrm{lin}}}{\partial \tilde{\boldsymbol{\mathcal{P}}}}$, then read as

$$\frac{\partial \boldsymbol{\mathcal{F}}^{\mathrm{lin}}_{ip}}{\partial \tilde{\boldsymbol{\mathcal{P}}}_{jq}} = \mu\nabla\Phi_{iq}\nabla\Phi_{jp} + \mu\delta_{pq}\sum_{l=1}^{3}\nabla\Phi_{il}\nabla\Phi_{jl} + \lambda\nabla\Phi_{ip}\nabla\Phi_{jq} = \frac{\partial \boldsymbol{\mathcal{F}}^{\mathrm{lin}}_{jq}}{\partial \tilde{\boldsymbol{\mathcal{P}}}_{ip}}\,. \tag{30}$$

A step-by-step derivation of this result can be found in the appendix 7.1.

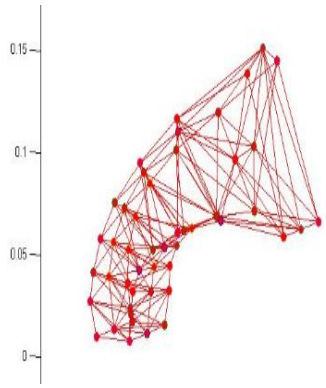
From the above equation (30) we see that in case the shape functions Φ are linear, their derivatives $\nabla\boldsymbol{\Phi}$ and hence the stiffness matrix $\partial\boldsymbol{\mathcal{F}}/\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}$ are constant on the whole element, which allows for crucial optimization of the algorithm. Complex parts of the time-stepping become suitable for precomputing at the beginning of the simulation, and can be omitted at run-time.

Furthermore, in FEM codes, the sparsity pattern of the system matrices corresponds to the elements' connectivity. A matrix block (i, j) is non-zero if and only if both nodes i and j share the same tetrahedron. A global matrix can thus be assembled by simply adding the contributions of the element matrices while considering the global node numbering. The resulting global matrix is a sparse, symmetric, positive definite matrix of the size $[3N \times 3N]$.

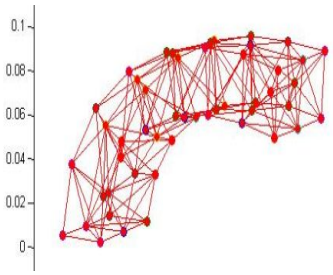
Even though applying the linearized force derivatives in the implementation allows for better computational efficiency, and thus, is of high importance for the practical computation in applications in everyday use, there is a substantial disadvantage: This is traded for accuracy. Especially when the deformations become significantly larger and the displacements grow too much, the simple application of the linear strain tensor $\boldsymbol{\varepsilon}^{\mathrm{lin}}$ does not produce satisfying results anymore. Unrealistic deformations become noticable when so-called '*ghost forces*' cause the deformed object to blow up unnaturally (see Figure 1a).

The reason for the unrealistic behaviour can be found in the generally non-linear systems of the elasticity problem and therefore in the fact that the elastic forces $\boldsymbol{F}(\boldsymbol{\mathcal{U}})$ as in (22) generally non-linearly depend on $\boldsymbol{\mathcal{U}}$, due to the actual geometric non-linearity of the strain tensor $\boldsymbol{\varepsilon}$ even if the response function C, is linear, i.e., represents a linear relation between stress and strain, which we assume in the linear elasticity theory. As soon as significant deformations occur, simply using the linear strain tensor $\boldsymbol{\varepsilon}^{\mathrm{lin}}$ does not produce satisfying results anymore.

Therefore, when simulating large displacements more advanced methods have to be applied. A very elegant solution is given by the '*corotational method*', which treats the rigid rotation and the deformation of the elements separately while still using the linear strain tensor $\boldsymbol{\varepsilon}^{\mathrm{lin}}$, and hence still being based on the above derived linearized version of the stiffness matrix. As opposed to the above example, the corotational method shows visually pleasent and realistic results, compare Figure 1b.



(a) The results of the simulation for the linearized version.



(b) The results of the simulation for the corotational version

Figure 1: Linearization- and Corotation-based Simulation results for the deformation of a beam under the effect of external traction forces and prescribed displacement boundary conditions.

3.7 The Corotational Method

The corotational method is based on the linear FEM with Cauchy's linear strain $\boldsymbol{\varepsilon}^{\text{lin}}$, however, aligns the elements with their reference configuration prior to the force computation. This means that the rotational part of the deformation is 'extracted', i.e., factored out in order to rotate the attached global coordinate frame into the reference frame, before an almost rigid transformation is applied and followed by the actual computation of the forces with respect to the non-rotated reference frame. The deformation forces computed for the aligned configuration are then rotated back to the current configuration afterwards.

The quality of such a *corotation*-based elasticity simulation is strongly influenced by the time efficiency and robustness of the estimation of the *rotation matrices* $\boldsymbol{R}$. In this work, we rely on the suggestion of Higham and Schreiber (compare [11]), who use the (right) polar decomposition of the deformation gradient $\nabla\boldsymbol{\varphi}$,

$$\nabla\boldsymbol{\varphi} = \boldsymbol{R}^T \nabla\boldsymbol{\varphi}^{\text{CR}}\,, \tag{31}$$

with the orthogonal rotation matrix $\boldsymbol{R}^T$, and a unique symmetric, positive definite (SPD) *rotation-free (co-rotated, CR) deformation gradient* or *stretch matrix* $\nabla\boldsymbol{\varphi}^{\text{CR}} = \boldsymbol{R}\nabla\boldsymbol{\varphi} = \nabla(\boldsymbol{R}\boldsymbol{\varphi})$. This yields stable and visually pleasing results. Since for a quadratic tetrahedron a single rotation matrix $\boldsymbol{R}$ is not enough to rotate it into a configuration that leaves a rotation-free deformation gradient $\boldsymbol{R}\nabla\boldsymbol{\varphi}$, a separate transformation and a separate polar decomposition of $\nabla\boldsymbol{\varphi}$ has to be applied for each cubature point.

For the calculation of the corotated force derivatives, we consider the strain and stress first. As can be seen in the following, the strain tensor $\boldsymbol{\varepsilon}$ fully cancels out the rotation

$$\begin{aligned}\boldsymbol{\varepsilon}\left(\nabla(\boldsymbol{R}\boldsymbol{\varphi})\right) = \boldsymbol{\varepsilon}(\nabla\boldsymbol{\varphi}^{\text{CR}}) &= \frac{1}{2}\left((\nabla\boldsymbol{\varphi}^{\text{CR}})^T(\nabla\boldsymbol{\varphi}^{\text{CR}}) - \boldsymbol{I}\right)\\ &= \frac{1}{2}\left(\left(\boldsymbol{R}\nabla\boldsymbol{\varphi}\right)^T\left(\boldsymbol{R}\nabla\boldsymbol{\varphi}\right) - \boldsymbol{I}\right) = \frac{1}{2}\left(\nabla\boldsymbol{\varphi}^T\boldsymbol{R}^T\boldsymbol{R}\nabla\boldsymbol{\varphi} - \boldsymbol{I}\right) = \boldsymbol{\varepsilon}(\nabla\boldsymbol{\varphi})\,.\end{aligned} \tag{32}$$

Hence, we conclude that the rotation-free deformation gradient allows for obtaining a small-strain approximation with the linearized strain tensor $\boldsymbol{\varepsilon}^{\text{lin}}$

$$\boldsymbol{\varepsilon}^{\text{lin}}(\nabla\boldsymbol{\varphi}^{\text{CR}}) = \boldsymbol{\varepsilon}^{\text{lin}}(\boldsymbol{R}\nabla\boldsymbol{\varphi}) = \frac{1}{2}\left(\boldsymbol{R}\nabla\varphi + \nabla\varphi^T\boldsymbol{R}^T\right) - \boldsymbol{I} =: \boldsymbol{\varepsilon}^{\text{CR}}(\nabla\boldsymbol{\varphi})\,. \tag{33}$$

Along with the rotation-invariant stress tensor

$$\boldsymbol{\sigma}^{\text{CR}}(\nabla\boldsymbol{\varphi})\,\boldsymbol{R}^T = \mathbb{C}\,\boldsymbol{\varepsilon}^{\text{CR}}(\nabla\boldsymbol{\varphi})\,\boldsymbol{R}^T = \mathbb{C}\,\boldsymbol{\varepsilon}^{\text{lin}}(\boldsymbol{R}\nabla\boldsymbol{\varphi})\,\boldsymbol{R}^T = \boldsymbol{\sigma}^{\text{lin}}(\boldsymbol{R}\nabla\boldsymbol{\varphi})\,\boldsymbol{R}^T = \boldsymbol{\sigma}^{\text{lin}}(\nabla\boldsymbol{\varphi}^{\text{CR}})\,\boldsymbol{R}^T \tag{34}$$

we find (according to (16)) that

$$\delta\boldsymbol{\varepsilon}^{\text{CR}} : \boldsymbol{\sigma}^{\text{CR}} = \delta\boldsymbol{\mathcal{U}} : \underbrace{\overbrace{\nabla\Phi\,\boldsymbol{\sigma}^{\text{lin}}}^{=\boldsymbol{\mathcal{F}}^{\text{lin}}(\nabla\boldsymbol{\varphi}^{\text{CR}})}\,\boldsymbol{R}^T}_{=:\boldsymbol{\mathcal{F}}^{\text{CR}}} = \delta\boldsymbol{\mathcal{U}} : \boldsymbol{\mathcal{F}}^{\text{CR}}\,, \tag{35}$$

which analogously to the above results suggests the shape function gradients $\nabla\Phi$ to be precomputed for every cubature point, and that

$$\boldsymbol{\mathcal{F}}^{\mathrm{CR}} = \boldsymbol{\mathcal{F}}^{\mathrm{lin}}\left(\nabla\boldsymbol{\varphi}^{\mathrm{CR}}\right)\,\boldsymbol{R}^T \overset{\nabla\boldsymbol{\varphi}^{\mathrm{CR}}=\boldsymbol{R}\nabla\boldsymbol{\varphi}^{\mathrm{lin}}}{=} \boldsymbol{R}\,\boldsymbol{\mathcal{F}}^{\mathrm{lin}}\,\boldsymbol{R}^T\,. \tag{36}$$

According to the derivation of the linearized version of the stiffness terms, we can now summarize the symmetric *'corotated force derivatives'* $\partial\boldsymbol{\mathcal{F}}^{\mathrm{CR}}/\partial\boldsymbol{\mathcal{U}}$, or $\partial\boldsymbol{\mathcal{F}}^{\mathrm{CR}}/\partial\tilde{\boldsymbol{\mathcal{P}}}$, respectively, i.e., the co-rotated stiffness terms,

$$\begin{aligned}\frac{\partial\boldsymbol{\mathcal{F}}^{\mathrm{CR}}_{ip}}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}} &= \sum_{k=1}^{3}\frac{\partial\boldsymbol{\mathcal{F}}^{\mathrm{C}}_{ik}(\nabla\boldsymbol{\varphi}^{\mathrm{CR}})}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}}\boldsymbol{R}^T_{kp} = \sum_{k=1}^{3}\boldsymbol{R}_{kp}\frac{\partial\boldsymbol{\mathcal{F}}^{\mathrm{C}}_{ik}(\nabla\boldsymbol{\varphi})}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}}\boldsymbol{R}^T_{kp}\\ &= \sum_{k=1}^{3}\boldsymbol{R}_{kp}\underbrace{\left(\mu\nabla\Phi_{iq}\nabla\Phi_{jk}+\mu\delta_{pk}\sum_{l=1}^{3}\nabla\Phi_{il}\nabla\Phi_{jl}+\lambda\nabla\Phi_{ik}\nabla\Phi_{jq}\right)}_{=\frac{\partial\boldsymbol{\mathcal{F}}^{\mathrm{lin}}_{ik}}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}}=:\boldsymbol{\mathcal{F}}^{\mathrm{lin}}_{i,j}}\boldsymbol{R}_{pk} = \frac{\partial\boldsymbol{\mathcal{F}}^{\mathrm{CR}}_{jq}}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{ip}}\,.\end{aligned} \tag{37}$$

Concluding, for the co-rotated element stiffness matrix ${}_{elem}\mathbb{K}^{\mathrm{CR}} = \nabla_{\boldsymbol{\mathcal{U}}}\boldsymbol{F}^{\mathrm{CR}} = \partial\boldsymbol{\mathcal{F}}^{\mathrm{CR}}/\partial\tilde{\boldsymbol{\mathcal{P}}}$, we find that the co-rotated gradient $\boldsymbol{\mathcal{F}}^{\mathrm{CR}}_{i,j}$ of the i-th nodal force ($i=1\ldots n$) in the direction of the j-th node $\tilde{\boldsymbol{\mathcal{T}}}_j$ ($j=1\ldots n$) reads as

$${}_{elem}\mathbb{K}^{\mathrm{CR}}_{ij} = \left(\nabla_{\boldsymbol{\mathcal{U}}}\boldsymbol{F}^{\mathrm{CR}}\right)_{i,j} = \boldsymbol{\mathcal{F}}^{\mathrm{CR}}_{i,j} = \boldsymbol{R}\,\boldsymbol{\mathcal{F}}^{\mathrm{lin}}_{i,j}\,\boldsymbol{R}^T \;\in\; \mathbb{R}^{3\times3}\,. \tag{38}$$

The fact that ${}_{elem}\mathbb{K}^{\mathrm{CR}}_{ij}$ only depends on the current rotation $\boldsymbol{R}$ and on the constant matrices $\boldsymbol{\mathcal{F}}^{\mathrm{lin}}_{i,j}$, obviously recommends the constant $[3\times3]$ blocks $\boldsymbol{\mathcal{F}}^{\mathrm{lin}}_{i,j}$ ($i,j=1\ldots n$) to be precomputed at the integration points and moreover to therein exploit symmetries, as stated for the linearized calculation above. This not only brings along higher efficiency, but also, due to corotation, it yields stable and realistic simulation results, for both linear and quadratic tetrahedra.

Finally, regarding the equilibrium equation (25) with respect to the whole (dynamic) system, we find that, using matrix-vector-notation again with $\boldsymbol{u} = \tilde{\boldsymbol{x}} - \boldsymbol{x} = (\boldsymbol{\varphi} - \mathrm{id})(\boldsymbol{x})$, it is transformed into

$$\mathbb{M}\ddot{\boldsymbol{u}} + \mathbb{D}\dot{\boldsymbol{u}} + \mathbb{K}(\tilde{\boldsymbol{x}} - \boldsymbol{x}) = \boldsymbol{f}_{ext}\,, \tag{39}$$

allowing us to retrieve the corotational version which reads as

$$\mathbb{M}\ddot{\boldsymbol{u}} + \mathbb{D}\dot{\boldsymbol{u}} + \mathbb{R}\,\mathbb{K}^{\mathrm{lin}}\,(\mathbb{R}^T\tilde{\boldsymbol{x}} - \boldsymbol{x}) = \boldsymbol{f}_{ext}\,, \tag{40}$$

therein linearizing the stiffness term.

4 Simulation of Cuts – Modelling Discontinuities using X-FEM

In the field of medical engineering, there is a wide range of simulation applications that benefit from biomechanical modeling of soft tissue, however, in particular the capability to simulate surgical cuts, which is an essential feature of a present-day medical simulation system, still is a challenging task. Even though a number of different approaches has been presented, the problems have not yet been solved satisfyingly, meaning that most of the recently developed methods suffer from both performance and stability issues. As opposed to this, the *Extended Finite Element Method* (X-FEM) seems to be very promising, as it effectively models discontinuities within an FEM mesh without creating new mesh elements and thus significantly reduces the impact on the performance of the simulation.

4.1 Different Approaches to Model Discontinuities

Most current methods for simulating the cutting of virtual objects published so far require the objects' elements to align with the cut. In order to achieve this *alignment* there are many options, however, even the most representative approaches reveal difficulties in their application.

A very basic and intuitive method simply removes the elements which were touched or intersected by the cutting surface from the mesh. New elements are not created, so this method does not harm the simulation's stability, however, it violates the mass conservation law since the open space is not filled anymore, compare [8].

Other methods suggest the cutting surface to be constrained to the borders of existing elements in the body, or – the opposite way around – nodes of existing elements to be snapped to the cut's trajectory, see [17]. The latter one of these approaches does not create new elements either and therefore keeps the computation simple, however, the mesh topology needs to be updated if the snapping distance becomes too large. Opposed to this, constraining the cut to element borders most often does not satisfy accuracy requirements since the cut is not actually going through where it should go, and thus produces projection errors.

Several research papers therefore suggest a subdivision of the elements that are intersected by the cutting surface, e.g., [14]. The algorithm is typically broken down into three simpler steps: element removal, element subdivision, and element addition. These, however, fundamentally modify the object's mesh structure. As a consequence, there are not only computationally expensive topology updates to be executed, but also, the increased number of element nodes leads to an extended dimension of the system matrices, which have to be reassembled with respect to the new topology. Yet, the main drawback of this method is the creation of possibly ill-shaped, ill-conditioned elements (slivers) that cause numerical instability in the simulation.

Apart from these approaches, Ted Belytschko and his colleagues introduced *meshless methods* using a set of particles without a fixed neighbourhood relationship to represent the simulated object. Those methods allow topological changes, however, they are computationally more expensive than FEM, and require additional structures to identify topologically separated particles, see [3].

4.2 The X-FEM Approach to Model Discontinuities

Most of the classical remeshing-based methods suffer from stability problems when a small part of a tissue is separated. The reasons for the instability are twofold: firstly, depending on the specific way of subdividing the element, remeshing can create ill-shaped sub-elements, and secondly, the masses of the new elements can become too low, such that mass ratios vanish and the matrices become singular. Simulation results hence are biased. As opposed to that, the X-FEM approach does not require remeshing. The cut – represented as a discontinuity – is modelled by means of additional discontinuous nodal *enrichment functions*. Moreover, the choice of an appropriate mass-lumping technique helps to stabilize dynamic simulation, regardless of the cut location.

The Extended Finite Element Method (X-FEM) is a numerical technique that extends the classical FEM approach by extending the solution space for solutions to differential equations with discontinuous functions. It was developed in 1999 by Ted Belytschko and collaborators, to help alleviate shortcomings of the FEM, and has been used, in particular, to model the propagation of both strong and weak discontinuities, meaning, e.g., cracks or material interfaces. This paragraph, which strongly builds on the work of [12], is to give a short introduction into the X-FEM, and we will show its suitability for cutting deformable objects in virtual environments. For more information on the X-FEM, we refer to [9] or a set of other relevant papers and books.

The basic idea of the X-FEM consists in enriching the elements by splitting their basis functions along the desired discontinuity. This effectively doubles the elements' degrees of freedom (DOFs) and decouples the solutions on either side of the discontinuity. In an elasticity simulation, this allows a single element to be cut or fractured into two (or more) independent parts. The X-FEM therefore modifies the functional representation on a subelement level, as clearly opposed to the above mentioned methods, which are changing the topology of the element mesh and also of the system matrices by means of complex remeshing and subdivision of existing elements into smaller elements in order to resolve the geometry of the cut. Moreover, since it models discontinuities within an FEM mesh without creating new mesh elements, this implies a small impact on the performance of the simulation

Looking back to the standard FEM, displacement fields are generally approximated using a mesh of elements connected at nodes. The object's deformation is given by the displacement of the nodes according to external and internal forces. These forces balance each other to a certain degree, as from where a deformation compensates for all surplus forces. Shape functions $\Phi_i \in \mathbb{R}^3$ interpolate the deformation

$\boldsymbol{\varphi}(x) \in \mathbb{R}^3$ and the displacement $u \in \mathbb{R}^3$ within an element from the nodal displacements $\boldsymbol{u}_i \in \mathbb{R}^3$, such that the displacement $u(x) \in \mathbb{R}^3$ of an arbitrary point $x \in \mathbb{R}^3$ can be computed as

$$u(x) = \sum_{i=1}^{n} \Phi_i(x) \boldsymbol{u}_i \,, \tag{41}$$

with the number n of element nodes.

When a discontinuity, i.e., a crack or cut, is added, the surrounding mesh nodes are *enriched* by an additional *global, discontinuous enrichment function* multiplied by shape functions with a *local* support, thereby again leading to a *local* discontinuous enrichment. The corresponding number of nodal DOFs is added and the displacement $u(x)$ of an arbitrary point x within an element is now computed as

$$u(x) = \sum_{i=1}^{n} \Phi_i(x) \boldsymbol{u}_i + \sum_{j=1}^{n} \Phi_j^*(x) \Psi_j(x) \boldsymbol{a}_j \,, \tag{42}$$

where $\Psi_j \in \mathbb{R}^3$ is the discontinuous enrichment function, and $\boldsymbol{a}_j \in \mathbb{R}^3$ is an added nodal DOF. The shape functions of the added DOFs $\Phi_j^* \in \mathbb{R}^3$ do not necessarily need to be identical to the standard shape functions Φ_i of the corresponding nodes, such that, e.g., higher order shape functions Φ_i might be used with linear enrichment shape functions Φ_j^*, in order to better fit the current requirements. However, in this work and the appendant implementation, we let $\Phi_j^* = \Phi_i$. Finally, the Φ_j^* generally build a partition of unity in local parts of the domain, as the X-FEM is based on the concept of the *partition of unity method* (PUM).

The enrichment function $\Psi(x)$ can be any arbitrary discontinuous function, provided that it is discontinuous along the crack or cut domain. Arguably the simplest choice is given by the *(generalized) Heaviside function* $H(x)$ also known as the *sign*() function, which assumes the value of $+1$ on the one side of the cut and -1 on the other side

$$\Psi(x) = H(x) = \begin{cases} +1 & \text{above the crack} \\ -1 & \text{below the crack} \end{cases} . \tag{43}$$

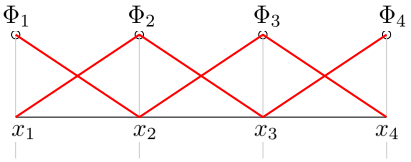
Thus, according to the above formula (42) for the calculation of the displacement of an arbitrary point x, we can now realize the splitting of the (original) basis functions Φ_i by adding the *enrichment basis functions* $\Phi_i \Psi_i$, weighted with their corresponding DOFs $\boldsymbol{a}_i$. However, the node enrichment impacts all elements sharing the enriched nodes – specifically, one ring of neighbours of the element containing the cut. Moreover, because of the additional term in (42), the shape functions together with the local enrichment functions generally do not have the *Kronecker delta property* anymore, and the enriched nodes' displacement is computed as a sum of the components $\boldsymbol{u}_i + \Psi_i \boldsymbol{a}_i$ as opposed to just $\boldsymbol{u}_i$.

Taking this disadvantage as a reason for enhancements, [20] proposes – instead of using the generalized Heaviside function $H(x)$ –, to apply the so-called *shifted enrichment functions* $\Psi_i(x)$, which are zero at all nodes and keep the enrichment local

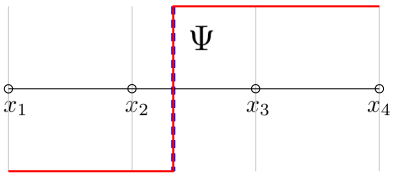
$$\Psi_i(x) = \frac{H(x) - H_i}{2} \,, \tag{44}$$

where H_i is the value of the generalized Heaviside function $H(x)$ at the i-th node, and the division by 2 ensures the Kronecker delta property. The shifted enrichment function Ψ_i vanishes at its associated node i, and the nodal basis function Φ_i vanishes at all nodes other than i, such that the enrichment basis functions $\Phi_i \Psi_i$ vanish at all nodal positions, and the displacement at each node i is fully defined by $\boldsymbol{u}_i$ alone. The contributions of $\boldsymbol{a}_i$ are only needed to determine the displacement within the cut (i.e., enriched) element, which also implies that the physical meaning of the $\boldsymbol{u}_i$ as the nodal displacements is restored. For the plotting of the simulation results, this is of high importance, and moreover, it simplifies the implementation of Dirichlet boundary conditions, as the displacement of a node can be fixed to a specific value by constraining a single DOF. Regarding the implementation, we introduce the concept of the so-called *level set method* later, which allows the definition of subspaces, which in terms of the cut object correspond to the two cut element parts above and below the cutting plane.

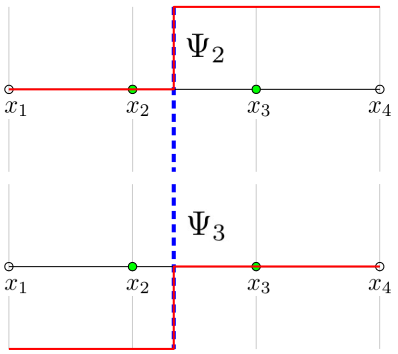
See Figure 2 for a 1D illustration of the two different enrichment approaches. On the left, the enrichment basis functions extend to all elements incident to the enriched node, as opposed to the right



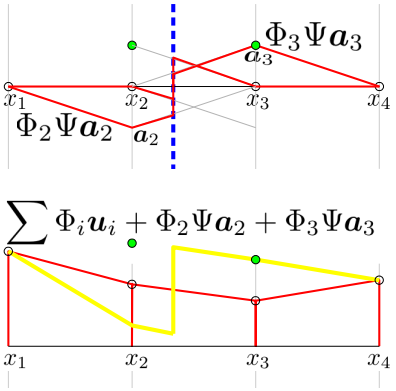
(a) Standard FEM: standard basis functions Φ_i.



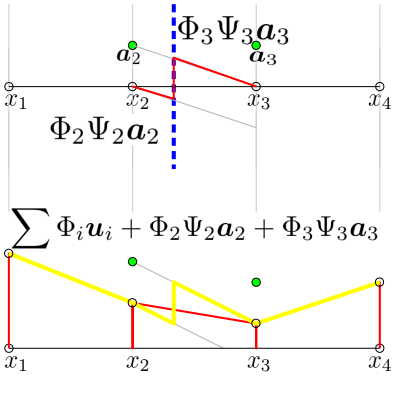
(b) X-FEM: Generalized Heaviside enrichment function $\Psi(x) = H(x)$.



(c) X-FEM: Shifted enrichment function $\Psi_i(x) = \frac{1}{2}(H(x) - H_i)$.



(d) Generalized Heaviside enrichment approach: product of standard shape functions and enrichment functions.



(e) Shifted enrichment approach: product of standard shape functions and enrichment functions.

Figure 2: The comparison of the two different enrichment approaches: using either the generalized Heaviside enrichment $\Psi(x) = H(x)$ (left column) or the shifted enrichment $\Psi_i(x) = \frac{1}{2}(H(x) - H_i)$ (right column).

side showing the shifted enrichment functions, the support of which comprises the cut element only. This locality simplifies the implementation and calculation enormously and also improves the stability of the simulation. Note that the shifted enrichment basis functions together with the original basis functions still span the same space as in the case of the unshifted enrichment functions.

The principle of enrichment functions is general, and thus can be used with any type of finite elements and any type of constitutive model. It allows the simulation of both strong and weak discontinuities. Strong discontinuities exist, e.g., as cuts or cracks, where parts of material are separated, whereas in a

weak discontinuity, such as in a material interface, the displacements are continuous, however, they contain a kink, making the displacement gradient discontinuous. By means of including optional enhancements such as a corotational formulation or mass lumping techniques, the X-FEM not only surpasses most currently used remeshing methods in both performance and stability, but also becomes suitable for very complex, interactive simulations of large deformations.

4.3 Simulation based on Linear and Corotational X-FEM

When an element is cut and a discontinuity added, the X-FEM doubles the number of element DOFs in order to decouple the solutions on both sides of the discontinuity. Hence, when assembling the system matrices and vectors, the increased number of DOFs has to be considered.

Based on the above notation in formula (42), the new DOFs are appended to the element displacement and force vectors $\boldsymbol{u}$ and $\boldsymbol{f}$:

$$ {}_{elem}\boldsymbol{u}^X = [\boldsymbol{u}_1 \ \dots \ \boldsymbol{u}_n \ \boldsymbol{a}_1 \ \dots \ \boldsymbol{a}_n]^T \ , \tag{45} $$

$$ {}_{elem}\boldsymbol{f}^X = [\boldsymbol{f}_1 \ \cdots \ \boldsymbol{f}_n \ \boldsymbol{f}_1^a \ \cdots \ \boldsymbol{f}_n^a]^T \ , \tag{46} $$

where the $\boldsymbol{u}_i$ represent the *usual* standard nodal DOFs (i.e., the standard FEM nodal displacements), and the $\boldsymbol{a}_i$ denote the *additional* nodal DOFs.

Concerning the system matrices, we found that a $[3n \times 3n]$ standard FEM element stiffness matrix ${}_{elem}\mathbb{K}$ is computed by adding up the n^2 $[3 \times 3]$ components ${}_{elem}\mathbb{K}_{ij}$ with respect to their node numbering, as in (27). Using Voigt notation, the components ${}_{elem}\mathbb{K}_{ij}$ can be retrieved as

$$ {}_{elem}\mathbb{K}_{ij} = \int_V \mathbb{B}_i^T \mathbb{C} \mathbb{B}_j \, dV \, , \tag{47} $$

where $\mathbb{C}$ denotes the $[6 \times 6]$ *stress-strain material matrix* as in (6), and the $\mathbb{B}_i$ represent the $[6 \times 3]$ *strain-displacement matrices*, which – by means of the shape function derivatives – account for the element geometry and the chosen strain measure. The indices i and j refer to the respective element nodes.

Assuming a static problem, i.e., regarding the time-independent versions of (39) and (40), this allows us, for the linearized version, to calculate the components of the force vector as

$$ \boldsymbol{f}_i = \sum_{j=1}^{n} {}_{elem}\mathbb{K}_{ij} \boldsymbol{u}_j = \sum_{j=1}^{n} \left(\int_V \mathbb{B}_i^T \mathbb{C} \mathbb{B}_j \, dV \right) \boldsymbol{u}_j \, , \tag{48} $$

or for the corotational version as

$$ \boldsymbol{f}_i = \mathbb{R} \sum_{j=1}^{n} {}_{elem}\mathbb{K}_{ij}^{\mathrm{CR}} (\mathbb{R}^T \boldsymbol{x}_j - \boldsymbol{x}_{0_j}) = \mathbb{R} \sum_{j=1}^{n} \left(\int_V \mathbb{B}_i^T \mathbb{C} \mathbb{B}_j \, dV \right) (\mathbb{R}^T \boldsymbol{x}_j - \boldsymbol{x}_{0_j}) \, , \tag{49} $$

respectively.

For the X-FEM, the structure of the system matrices of the standard FEM can simply be transferred. Hence, using the *enriched strain-displacement matrix* $\mathbb{B}^X$ given by

$$ \mathbb{B}^X = [\, \mathbb{B}_1 \ \dots \ \mathbb{B}_n \ \Psi_1 \mathbb{B}_1 \ \dots \ \Psi_n \mathbb{B}_n \,] \ , \tag{50} $$

we accordingly find

$$ {}_{elem}\mathbb{K}_{ij}^X = \int_V \mathbb{B}_i^{XT} \mathbb{C} \mathbb{B}_j^X \, dV \, , \tag{51} $$

for the $[3 \times 3]$ *components of the enriched stiffness matrix.*

Having a closer look at the matrix $\mathbb{B}^X$, when dealing with linear tetrahedra, we can further transform this equation. Due to the fact that the enrichment functions $\Psi_i(x)$ take on constant values over the domain above (V_a) and below (V_b) the cut plane, and only change values from one to another, we can split the above volume integral into two parts

$$ {}_{elem}\mathbb{K}_{ij}^X = \underbrace{\int_{V_a} \mathbb{B}_i^{XT} \mathbb{C} \mathbb{B}_j^X \, dV}_{\text{Volume above the cut plane}} + \underbrace{\int_{V_b} \mathbb{B}_i^{XT} \mathbb{C} \mathbb{B}_j^X \, dV}_{\text{Volume below the cut plane}} \ . \tag{52} $$

Next, according to the above representation of the X-FEM vectors ${}_{elem}\boldsymbol{u}^X$ and ${}_{elem}\boldsymbol{f}^X$, we also divide the X-FEM element stiffness matrix into four parts

$$ {}_{elem}\mathbb{K}^X = \begin{bmatrix} \mathbb{K}^{uu} & \mathbb{K}^{ua} \\ \mathbb{K}^{au} & \mathbb{K}^{aa} \end{bmatrix} \tag{53} $$

where $\mathbb{K}^{uu}$ corresponds to the standard FEM element matrix with the original DOFs, whereas $\mathbb{K}^{ua}$, $\mathbb{K}^{au}$ and $\mathbb{K}^{aa}$ correspond to the added DOFs.

Note that these specifications can cause problems when dealing with partial cuts. In case a cutting surface intersects the body, yet stops inside the element, the above analysis does not hold anymore, and a subdivision into subelements above and below the cut is not well-defined. In this work, however, we only consider complete cuts. Therefore, we let a cutting surface which partially intersects an element continue through it, such that the affected element is fully cut into two parts above and below the cutting plane. Of course, when given a higher refined mesh around the cutting front line, the thus caused variance from the original geometry of the cut gets smaller, such that as from a certain refinement level it can be neglected.

In what follows, we show the actual formulations of the mass and stiffness matrices for the linearized and corotational method, therein relying on the results of [13] and explaining possible future enhancements, such as the application of quadratic elements.

4.3.1 Linear X-FEM

After substitution of equation (50) into equation (52), and when thereby assuming a constant matrix $\mathbb{B}$, the element stiffness matrix components become

$$ \mathbb{K}_{ij}^{uu} = \mathbb{K}_{ij} \tag{54} $$

$$ \mathbb{K}_{ij}^{ua} = \left(\frac{V_a}{V} \Psi_{aj} + \frac{V_b}{V} \Psi_{bj} \right) \mathbb{K}_{ij} \tag{55} $$

$$ \mathbb{K}_{ij}^{au} = \left(\frac{V_a}{V} \Psi_{ai} + \frac{V_b}{V} \Psi_{bi} \right) \mathbb{K}_{ij} \tag{56} $$

$$ \mathbb{K}_{ij}^{aa} = \left(\frac{V_a}{V} \Psi_{ai} \Psi_{aj} + \frac{V_b}{V} \Psi_{bi} \Psi_{bj} \right) \mathbb{K}_{ij} \,. \tag{57} $$

Here, $\mathbb{K}_{ij}^{uu}$ denotes a matrix component of which none of the element nodes contributes with additional DOFs, $\mathbb{K}_{ij}^{ua}$ symbolizes a component of which node i contributes with its usual standard FEM part and node j with its additional X-FEM part, vice versa for $\mathbb{K}_{ij}^{au}$, and finally, $\mathbb{K}_{ij}^{aa}$ is a component where both nodes contribute with their additional DOFs. Moreover, Ψ_{ai} denotes the value of the function $\Psi_i(x)$ above the cut plane, and Ψ_{bi} stands for the function's value below the cut.

Note that $\mathbb{K}^{uu}$ is identical to the original element stiffness matrix of the non-enriched element, whereas for $\mathbb{K}^{ua}$, $\mathbb{K}^{au}$ and $\mathbb{K}^{aa}$ parts of the original stiffness matrix are multiplied by constant factors which only depend on the size of the dissected volumes and the cut side of the given node (above or below). This structure of the element matrices thus allows to essentially simplify their calculation, since when cutting an element, there is no need for additional integration points for the numerical volume integration, but instead the use of volume ratios will do. This is to be pointed out as a very special feature and will be discussed in detail in the implementation section.

A further enormous simplification is achieved when applying the *shifted enrichment*. Inserting

$$ \Psi_{aj} = \frac{\overbrace{H_{aj}}^{=+1} - H_j}{2} = \begin{cases} 0 & \text{if } H_j = H_{aj} = +1 \text{ above the crack and on the crack line} \\ +1 & \text{if } H_j = H_{bj} = -1 \text{ below the crack} \end{cases} \tag{58} $$

$$ \Psi_{bj} = \frac{\overbrace{H_{bj}}^{=-1} - H_j}{2} = \begin{cases} -1 & \text{if } H_j = H_{aj} = +1 \text{ above the crack and on the crack line} \\ 0 & \text{if } H_j = H_{bj} = -1 \text{ below the crack} \end{cases} \tag{59} $$

into the above equations (54 – 57) instead of the generalized Heaviside enrichment yields – after rearrangement – the following element stiffness matrix components

$$\mathbb{K}_{ij}^{uu} = \mathbb{K}_{ij} \tag{60}$$

$$\mathbb{K}_{ij}^{ua} = \begin{cases} -\frac{V_b}{V}\mathbb{K}_{ij} & \text{if } H_j = +1 \\ +\frac{V_a}{V}\mathbb{K}_{ij} & \text{if } H_j = +1 \end{cases} \tag{61}$$

$$\mathbb{K}_{ij}^{au} = \begin{cases} -\frac{V_b}{V}\mathbb{K}_{ij} & \text{if } H_i = +1 \\ +\frac{V_a}{V}\mathbb{K}_{ij} & \text{if } H_i = -1 \end{cases} \tag{62}$$

$$\mathbb{K}_{ij}^{aa} = \begin{cases} +\frac{V_b}{V}\mathbb{K}_{ij} & \text{if } Hi = H_j = +1 \\ +\frac{V_a}{V}\mathbb{K}_{ij} & \text{if } H_i = H_j = -1 \\ 0 & \text{if } H_i \neq H_j \end{cases} . \tag{63}$$

In our implementation, the computation of the volumes V, V_a, V_b is processed by simply adding up the weighted determinants of the concerned subelements, respectively above or below the cutting surface. Note that these results hold for quadratical elements, too, thus allowing for a simple adjustment as soon as the cutting surface inside the elements is determined. For an analysis of the computational overhead of an enriched element over a standard element, we refer to [12].

As mentioned before, the linear FEM is the simplest FE model as the stiffness matrix remains constant during the simulation. For X-FEM applications, the same feature applies: the enriched stiffness matrix can be computed once the position of the discontinuity is known and it remains constant afterwards, which is of great value with respect to computation time. However, as for the standard FEM, in the linear X-FEM case the annoying phenomenon of *ghost forces* also occurs when cuts and cracks grow wider and displacements become too large. In particular, when a cut element is rotated or when parts of an object are dissected and fall apart this leads to disturbing artefacts, which is why more sophisticated methods such as the corotation-based X-FEM must be applied.

4.3.2 Corotational X-FEM

In the corotational formulation, the linearization-based computation of the deformation is extended by the idea of separating a rigid rotation from the actual deformation. However, the application of the corotational formulation in an X-FEM simulation requires an additional essential circumstance to be considered: When an element is cut, the aligning rotations for the two parts above and below the cutting surface are obviously different, which hence requires two rotation matrices to be computed, compare Figure 3.

As presented in the implementation chapter, our algorithm subdivides a tetrahedral element's two (with the exception of plane cutting surfaces) arbitrarily shaped subelements on either side of the cut, such that they yield a set of tetrahedral subelements again. Hence, we actually have to compute the rotation matrices for all these subelements separately, accounting for the fact that we integrate over each of these subelements separately, too. However, for reasons of simplicity, we neglect this in the below notation and handle two (sub)elements only (one on either side of the cut), each of which has one integration point.

The deformation forces (compare equation (49) for the standard FEM) in an enriched element are hence computed as follows

$$\boldsymbol{f}_i^X = \sum_{j=1}^{n} \underbrace{\mathbb{R}_a \left(\int_{V_a} \mathbb{B}_i^{XT} \mathbb{C} \mathbb{B}_j^X \, dV \right) \left(\mathbb{R}_a^T \boldsymbol{x}_j^X - \boldsymbol{x}_{0_j}^X \right)}_{\text{above the cut plane}} + \sum_{j=1}^{n} \underbrace{\mathbb{R}_b \left(\int_{V_b} \mathbb{B}_i^{XT} \mathbb{C} \mathbb{B}_j^X \, dV \right) \left(\mathbb{R}_b^T \boldsymbol{x}_j^X - \boldsymbol{x}_{0_j}^X \right)}_{\text{below the cut plane}} , \tag{64}$$

where $\mathbb{R}_a$ and $\mathbb{R}_b$ are the rotations of the element parts above and below the cut plane, respectively, and the X-FEM vector $\boldsymbol{x}^X = \boldsymbol{x}_0^X + \boldsymbol{u}^X$. Inserting the generalized Heaviside enrichment functions, we find that the deformation forces for the standard DOFs ($0 \leq i \leq n$) read as

$$\boldsymbol{f}_i = \frac{V_a}{V} \mathbb{R}_a \sum_{j=1}^{n} \mathbb{K}_{ij} \left(\mathbb{R}_a^T \boldsymbol{x}_{aj} - \boldsymbol{x}_{0_j} \right) + \frac{V_b}{V} \mathbb{R}_b \sum_{j=1}^{n} \mathbb{K}_{ij} \left(\mathbb{R}_b^T \boldsymbol{x}_{bj} - \boldsymbol{x}_{0_j} \right) , \tag{65}$$

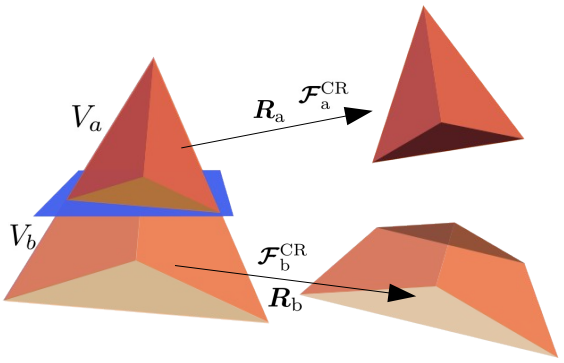


Figure 3: The dissected parts V_a, V_b of an element undergo generally different rotations $\boldsymbol{R}_a$, $\boldsymbol{R}_b$ and deformations $\boldsymbol{\mathcal{F}}_a^{\mathrm{CR}}$, $\boldsymbol{\mathcal{F}}_b^{\mathrm{CR}}$. The displacement of an arbitrary point within an element can be determined using equation (42). Especially, all points that are above or below the cut in the (non-deformed) reference configuration will remain on the same side of the cut in the deformed state, which also allows to determine the rotation matrices $\boldsymbol{R}_a$, $\boldsymbol{R}_b$.

where

$$\boldsymbol{x}_{aj} = \boldsymbol{x}_{0_j} + \boldsymbol{u}_j + \Psi_{aj}\boldsymbol{a}_j \quad \text{and} \quad \boldsymbol{x}_{bj} = \boldsymbol{x}_{0_j} + \boldsymbol{u}_j + \Psi_{bj}\boldsymbol{a}_j\,, \tag{66}$$

was derived from $\boldsymbol{x}_j = \boldsymbol{x}_{0_j} + \boldsymbol{u}_j + \Psi_j\boldsymbol{a}_j$.

Accordingly, for the additional DOFs $(n+1 \leq i \leq 2n)$ the deformation forces read as

$$\boldsymbol{f}_i^a = \frac{V_a}{V}\mathbb{R}_a\Psi_{ai}\sum_{j=1}^{n}\mathbb{K}_{ij}\left(\mathbb{R}_a^T\boldsymbol{x}_{aj} - \boldsymbol{x}_{0_j}\right) + \frac{V_b}{V}\mathbb{R}_b\Psi_{bi}\sum_{j=1}^{n}\mathbb{K}_{ij}\left(\mathbb{R}_b^T\boldsymbol{x}_{bj} - \boldsymbol{x}_{0_j}\right)\,. \tag{67}$$

Again, applying shifted enrichment yields essentially simplified results for the deformation forces. For a standard DOF $(0 \leq i \leq n)$, we retrieve

$$\boldsymbol{f}_i = \frac{V_a}{V}\mathbb{R}_a\sum_{j=1}^{n}\mathbb{K}_{ij}\left(\mathbb{R}_a^T\boldsymbol{x}_{aj} - \boldsymbol{x}_{0_j}\right) + \frac{V_b}{V}\mathbb{R}_b\sum_{j=1}^{n}\mathbb{K}_{ij}\left(\mathbb{R}_b^T\boldsymbol{x}_{bj} - \boldsymbol{x}_{0_j}\right)\,, \tag{68}$$

with

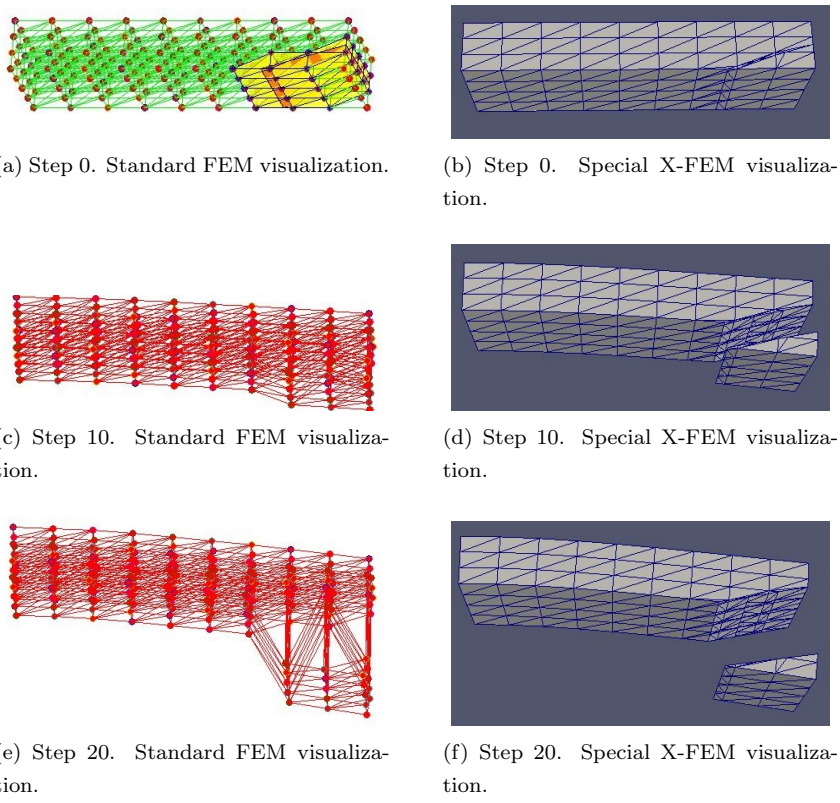
$$\boldsymbol{x}_{aj} = \begin{cases} \boldsymbol{x}_{0_j} + \boldsymbol{u}_j & \text{if } H_j = +1, \text{ i.e., if } \Psi_{aj} = 0 \\ \boldsymbol{x}_{0_j} + \boldsymbol{u}_j + \boldsymbol{a}_j & \text{if } H_j = -1, \text{ i.e., if } \Psi_{aj} = +1 \end{cases} \tag{69}$$

$$\boldsymbol{x}_{bj} = \begin{cases} \boldsymbol{x}_{0_j} + \boldsymbol{u}_j - \boldsymbol{a}_j & \text{if } H_j = +1, \text{ i.e., if } \Psi_{bj} = -1 \\ \boldsymbol{x}_{0_j} + \boldsymbol{u}_j & \text{if } H_j = -1, \text{ i.e., if } \Psi_{bj} = 0 \end{cases} \tag{70}$$

For the added DOFs $(n+1 \leq i \leq 2n)$, we find

$$\boldsymbol{f}_i^a = \begin{cases} -\frac{V_b}{V}\mathbb{R}_b\sum_{j=1}^{n}\mathbb{K}_{ij}\left(\mathbb{R}_b^T\boldsymbol{x}_{bj} - \boldsymbol{x}_{0_j}\right) & \text{if } H_i = -1, \text{ i.e., if } \Psi_{ai} = 0 \text{ and } \Psi_{bi} = -1 \\ +\frac{V_a}{V}\mathbb{R}_a\sum_{j=1}^{n}\mathbb{K}_{ij}\left(\mathbb{R}_a^T\boldsymbol{x}_{aj} - \boldsymbol{x}_{a_j}\right) & \text{if } H_i = +1, \text{ i.e., if } \Psi_{ai} = +1 \text{ and } \Psi_{bi} = 0 \end{cases}\,. \tag{71}$$

The hence resulting corotation-based X-FEM simulation produces satisfying and versatile results, which are not only stable, computationally efficient, and accurate with respect to the simulation error compared to benchmark results, but also visually pleasant, even for application scenarios that include large deformations, see Figure 4.



(a) Step 0. Standard FEM visualization.

(b) Step 0. Special X-FEM visualization.

(c) Step 10. Standard FEM visualization.

(d) Step 10. Special X-FEM visualization.

(e) Step 20. Standard FEM visualization.

(f) Step 20. Special X-FEM visualization.

Figure 4: Dissection of a beam. A complete cut is performed and the effects of gravity make the slice fall apart. The left side shows the original elements using a standard FEM visualization. The right side shows the separated elements as represented when using a special X-FEM visualization. Note: When applying the simpler linear X-FEM, disturbing ghost forces would blow up the dissected parts, especially when gravity causes the dissected slice to rotate while falling.

4.3.3 Assembly of global matrices

So far, only the structure of the (local) element X-FEM matrices and vectors with respect to a single enriched element has been addressed. In the following, we give some notes on their global structure, for what it is necessary to first determine the object's total number of enriched elements, i.e., the number of elements that are intersected by the cutting surface. For this purpose, a so-called *level set function* is introduced. In the implementation, we make use of the signed-distance function Λ given by

$$\Lambda(\boldsymbol{x}) = \pm \min_{\boldsymbol{x}^* \in \Gamma_{cut}} \| \boldsymbol{x} - \boldsymbol{x}^* \|, \quad \boldsymbol{x} \in \Omega, \tag{72}$$

where $\| \cdot \|$ denotes the Euclidean norm, $\boldsymbol{x}^*$ is the closest point on the cut surface Γ_{cut} to $\boldsymbol{x}$, and Ω is the object's domain. For a discretized domain, the values of the level set function are typically stored at the nodes $\Lambda_i = \Lambda(\boldsymbol{x}_i)$, where $i = 1, \ldots, N$. In this way, the level set method not only allows to define interfaces implicitly by means of the zero-level of a scalar function within the domain, but also facilitates the construction of enrichment as follows: Whether or not an element is cut by the discontinuity is determined on the element-level by means of the level set function Λ. The set of cut elements is

$$M_\Lambda = \left\{ k \in \{1, \ldots, N_{el}\} : \min_{i \in I_k} (\Lambda(\boldsymbol{x}_i)) \cdot \max_{i \in I_k} (\Lambda(\boldsymbol{x}_i)) < 0 \right\}, \tag{73}$$

where N_{el} denotes the total number of elements in the domain Ω, and I_k stands for the element nodes of the k-th element. In other words, the implementation is based on a simple if-else decision, computing

whether or not the considered element's nodes lie on different sides of the cut, i.e., are afflicted with different signs (± 1) by the signed-distance function Λ. Subsequently, based on the determination of the enriched elements, a vector is set up, which contains the enriched nodes; each node listed only once. From the number of additional DOFs, we can derive the structure of the global X-FEM vectors and matrices. A global X-FEM vector then reads as

$$ {}_{global}\boldsymbol{u}^X = [\boldsymbol{u}_1 \ \ldots \ \boldsymbol{u}_N \ \boldsymbol{a}_1 \ \ldots \ \boldsymbol{a}_{N_{enr}}]^T \ , \tag{74} $$

where, again, N denotes the object's total number of nodes, and N_{enr} identifies the total number of enriched nodes in the object, with $N_{enr} \leq N$. Note that since we work in 3D space each entry actually consists of three components. As mentioned before, when applying shifted enrichment functions, the nodal displacements of both enriched and non-enriched nodes are stored directly in the $\boldsymbol{u}_i$, whereas the contributions of the $\boldsymbol{a}_i$ are only needed to determine the displacement within an enriched element, such that the physical meaning of the nodal displacements is remains preserved. A global X-FEM matrix can again be divided into four submatrices as follows

$$ {}_{global}\mathbb{K}^X = \begin{bmatrix} \mathbb{K}_1^X & \mathbb{K}_3^X \\ \mathbb{K}_2^X & \mathbb{K}_4^X \end{bmatrix} , \tag{75} $$

where the dimensions of the submatrices correspond to the vectors' dimensions: $\mathbb{K}_1^X \in \mathbb{R}^{(3N \times 3N)}$ corresponds to the standard FEM matrix with the standard DOFs, whereas $\mathbb{K}_2^X \in \mathbb{R}^{(3N_{enr} \times 3N)}$, $\mathbb{K}_3^X \in \mathbb{R}^{(3N \times 3N_{enr})}$ and $\mathbb{K}_4^X \in \mathbb{R}^{(3N_{enr} \times 3N_{enr})}$ correspond to the additional DOFs.

4.4 Dynamical Simulation

In order to dynamically simulate the behaviour of soft tissue after a cut has been fully executed, the biomechanical model has to account for inertia and energy dissipation, and hence must consider mass and damping factors, as alredy explained for the standard FEM.

Similarly to the enriched element stiffness matrix, the enriched element mass matrix has the form

$$ {}_{elem}\mathbb{M}^X = \begin{bmatrix} \mathbb{M}^{uu} & \mathbb{M}^{ua} \\ \mathbb{M}^{au} & \mathbb{M}^{aa} \end{bmatrix} . \tag{76} $$

The components of the *consistent* element mass matrix of an enriched element are defined as

$$ \mathbb{M}_{ij}^{uu} = \rho \int_V \Phi_i \Phi_j \, dV \, , \tag{77} $$

$$ \mathbb{M}_{ij}^{ua} = \rho \int_V \Phi_i \Phi_j \Psi_j \, dV \, , \tag{78} $$

$$ \mathbb{M}_{ij}^{au} = \rho \int_V \Psi_i \Phi_i \Phi_j \, dV \, , \tag{79} $$

$$ \mathbb{M}_{ij}^{aa} = \rho \int_V \Psi_i \Phi_i \Phi_j \Psi_j \, dV \, . \tag{80} $$

Yet, as mentioned before for the standard FEM, problems during the simulation usually arise regarding the numerical stability. Most often, small sliver elements falling apart from the object have a negative impact on the numerical accuracy and stability of both explicit and implicit methods. When applying explicit methods, the choice of a proper *mass-lumping technique* can essentially improve simulation stability, which is why a number of different techniques have been developed. Generally, the lumped submatrices $\mathbb{M}_{ij}^{ua}$ and $\mathbb{M}_{ij}^{au}$ are zero, since they do not share any entries with the element matrices' diagonal. Opposedly, for $\mathbb{M}_{ij}^{uu}$ and $\mathbb{M}_{ij}^{aa}$ we need to distinguish between different lumping techniques. The most well-known technique achieves diagonalization by means of row summation. Alternatively, there are weighted diagonalization techniques. A third type of mass lumping techniques specific to the X-FEM is presented in [12], proposing

$$ \overline{\mathbb{M}}_{ii}^{uu} = \frac{m}{n} \tag{81} $$

$$ \overline{\mathbb{M}}_{ii}^{aa} = \left(\frac{V_a}{V} \Psi_{ai}^2 + \frac{V_b}{V} \Psi_{bi}^2 \right) \overline{\mathbb{M}}_{ii}^{uu} \tag{82} $$

where n again denotes the number of element nodes. A stability analysis with respect to different mass lumping techniques for an enriched element using the shifted enrichment function can be found in [12].

Summarizing, the X-FEM and our implementation along with the underlying data structures proved their suitability for cutting deformable objects in a virtual environment. Most of the drawbacks of other approaches to model discontinuities are surpassed in both performance and stability. In our implementation, an interactive performance of a progressive cut during run-time is not yet possible, as this would include a permanent updating process of the element structure and of the system's reaction to external and internal forces changing in time. As opposed to this, our implementation performs a dynamic simulation of the soft tissue behaviour after a cut has been fully executed, which is optimized in terms of reducing the computational costs during the actual time-stepping part in the simulation by means of executing specific parts of the calculation of the system matrices in precomputing steps. However, referring to [12], the X-FEM shows to be suitable even for interactive real-time virtual reality simulations. The next chapter will spotlight implementation and data structure of the simulation, whereafter an evaluation of the simulation results will follow.

5 Notes on Implementation and Data Structures

Starting from an existing 2D static FEM implementation[4] in MATLAB[5] for the simulation of the behaviour of an elastic body, we developed a 3D dynamic, corotation-based simulation, which allows for modelling the behaviour of soft tissue – in particular in the medical context – under the effect of external and internal forces. Both linear (Tet4) and quadratic (Tet10) elements furnish the implementation, providing better performance and higher accuracy in the simulation. In a subsequent step, the simulation capabilities were extended by means of incorporating the X-FEM in order to allow the simulation of surgical cuts.

Below, we will highlight the most important aspects of our MATLAB implementation prototype and the underlying data structures. Additional information can be found in the source code.

5.1 FEM Implementation

Given the standard structure of a general-purpose FEM code, in our simulation prototype we particularly focus on providing computational efficient and accurate algorithms aiming for near-to-real-time simulations. There is a great potential for optimization, whereof we consider the following points:

Assembly of the System Matrices. As mentioned above, complex parts of the time-stepping can be precomputed at the beginning of the simulation, especially with respect to reducing the overhead whenever a new stiffness matrix is needed. For instance, if the shape functions Φ are linear, their derivatives and the stiffness matrix are constant on the whole element, such that their calculation can be omitted at run-time. When applying the corotational method, we find that the current step's stiffness matrix only depends on the current rotation and constant stiffness matrices, the latter of which are therefore precomputed at the cubature points. Moreover, using a nodal basis as introduced above involves specific sparsity patterns of the system matrices, which correspond to the element's connectivity. An efficient volume integration that maps arbitrary (real) elements into local (reference) unit elements additionally accelerates the precomputing part by allowing to easily obtain all local geometric information by simply scaling mapped volume information.

Implicit Time Integration. In order to make the simulation feasible for real-time applications, we combine the corotation-based algorithm with the implicit Newmark time integration scheme, which enforces the simulation's almost unconditional stability. For an implementation of the Newmark algorithm we refer to a pseudo-code excerpt in [1].

[4] SolidMechanics.org offers useful information on the FEM implementation as well as sample FE codes on: http://solidmechanics.org/.

[5] MATLAB is a numerical computing environment and fourth-generation programming language, see http://www.mathworks.de/products/matlab/.

Incorporation of Boundary Conditions (BCs). While surface traction forces (natural Neumann BCs) are already incorporated in the weak formulation (13), geometric constraints (Dirichlet BCs) have to be considered separately. Common techniques in engineering include *Lagrangian Multipliers* [19] which link additional algebraic equations to the system, or *Penalty Methods* [21] which add additional forces on individual nodes. We build on directly including the Dirichlet boundary nodes into the right hand side (force) vector of the system, and correspondingly adapting the remaining system.

5.2 X-FEM Implementation

Generally, there are three major differences in an X-FEM code as compared to a classical FEM code: First, the cubature has to consider the special character of the enrichment, second, the enrichment functions have to be implemented and incorporated, and third, the code must be able to deal with a variable number of DOFs per node, which holds on the element-level (the element matrices have different dimensions) as well as for the overall system matrix (which is of the dimension $[3N + 3N_{enr}]$). Furthermore, the visualization has to be adjusted in order to allow the representation of inner-element discontinuities. In the following we describe an implementation for linear tetrahedra, yet the code is kept flexible for future extensions to quadratic tetrahedra and other element types. Subsequently, these items will be described with respect to their order in the whole simulation workflow.

1. Preprocessing Steps. Given an object's tetrahedralized geometry as well as material properties for each element, the simulation input consists of displacement (Dirichlet) and force (Neumann) BCs and the definition of a *cut*, represented by a simple 2D plane in 3D space. Currently, the cutting plane is implicitly given by means of the level set method, allowing for the calculation of the *level set values* of all nodes, i.e., the oriented distances between the cutting plane and the object's nodes. Future enhancements might allow the definition of a more flexible cutting surface, e.g., represented by formulations based on differential geometry.

Considering a situation where the local enrichment functions have jumps within the elements, the standard Gauss cubature does not deliver satisfying results anymore, since it requires smoothness of the integrands. For this reason, the X-FEM requires a *decomposition of the elements into subelements* that align with the discontinuity. Each of these subelements requires a separate integration, which involves new integration points in the subelements, as can be seen in Figure 5.

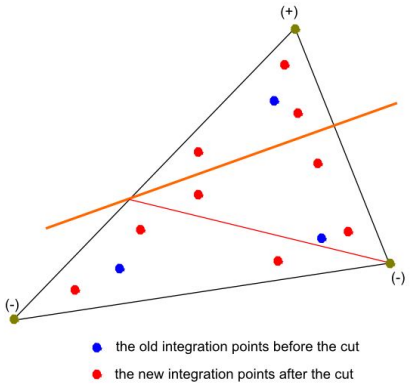
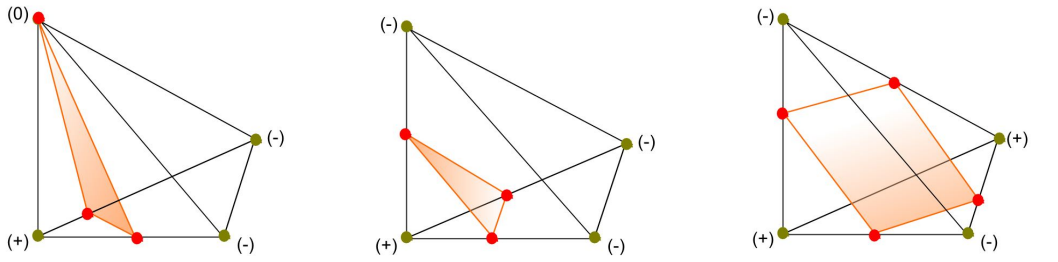


Figure 5: The distribution of the integration points in an element (2D linear triangular element) before and after a cut. The orange line represents the cut, the vertices above the cut are denoted by $(+)$, the ones below the cut by a $(-)$. The blue points denote the integration points in the original element before the cut. The red points illustrate the distribution of the integration points in the subelements resulting from the subdivision of the original element after the cut.

The actual element subdivision is based on the determination of the intersection points of the cutting plane and the element edges. In case of linear elements this step simply follows the rules of the intercept

theorems using the level set values of the current element's nodes. In case of quadratic or higher order elements it becomes more complex and computationally expensive, too.

Once the intersection points are known, the *element subdivision process* has to be started: First, the respective element is mapped into a reference element as explained above. Second, a *case-by-case analysis*, differentiating according to the values of the level set function in the original element nodes, is responsible for subdividing the element into two subelements, one above the cutting plane and one below it. Figure 6 shows 3 examples out of 8 different ways in which a tetrahedral element can be cut, 4 of which actually leave the element only touched along an edge or a face, thus "uncut".



(a) Case $(0+--)$ / $(0-++)$. Cut through one vertex and two opposite edges.

(b) Case $(+---)$ / $(-+++)$. Cut through three element edges.

(c) Case $(++--)$ / $(--++)$. Cut through four element edges.

Figure 6: Case-by-case analysis as implemented in the simulation. It shows the different ways in which a tetrahedral element can be cut. We denote the vertices above the cutting plane by $(+)$, those below it by $(-)$, and the nodes which are directly intersected by the cut by (0), according to the sign of the level set values. Hence, the cases can be identified by quadruples of these symbols, such that, e.g., $(+-0-)$ denotes an element where node 1 lies above the cut, nodes 2 and 4 lie below it, and node 3 lies on the cut. Of course, any arbitrary permutation is possible.

Obviously, the two subelements obtained from this subdivision process are not necessarily tetrahedra anymore, which is why we choose to again subdivide these subelements until they represent smaller tetrahedra, such that on the subelement-level we can apply the same functions on them as on the element-level.

Note that in order to prevent the element subdivision process from producing ill-conditioned elements which cause numerical instability, we define a minimum distance between an element node and a cut, until which we define the element not to be cut.

The result of the element subdivision process is a set of new subelements above and below the cut, which then have to be occupied by integration points and the respective integration weights, again based on the above-mentioned mapping on a reference element. This allows us to finally compute the element and subelement stiffness blocks ${}_{elem}\mathbb{K}_{ij}^{X}$ under consideration of the cut in a last preprocessing step before the actual simulation. When applying linear shape functions, which results in constant stiffness matrices, since the shape function derivatives are constant, and using shifted enrichment functions, this allows for maximally reducing the simulation overhead, which leads to a highly-optimized performance.

In general, considering the assembly of the system matrices, we notice that our X-FEM code must be able to deal with a variable number of DOFs per node, which holds both, on the element-level since the element matrices have different dimensions ($[3n \times 3n]$ or $[2 \cdot 3n \times 2 \cdot 3n]$, depending on whether or not the element is enriched), as well as for the overall system matrix (which is of the dimension $[3N + 3N_{enr} \times 3N + 3N_{enr}]$).

Hence, the above presented features require a number of adaptations and extensions to the standard FEM data structure. In the following, we therefore list our new *X-FEM data structure* and the respective additional entities with respect to the element level:

- **double** `LevelSetValues`: A vector containing the current element's nodal level set function values.

- **boolean** `ElementEnrichedYesNo`: A boolean variable indicating whether or not the element is cut, i.e., intersected by the cutting plane, and hence must be enriched. According to the above-described case differentiation, it moreover differentiates between an element that is just touched (in a vertex, edge or face) and an element which is really cut through.
- **int** `NumberOfSubelements`: A scalar variable, which is calculated by the element subdivision algorithm, which indicates an element's number of subelements.
- **double** `SubelementCoordinates`: A matrix containing the coordinates of the subelements of the current element.
- **array** `SubelementIntegration`: An array containing information about the integration over the subelements, such as the number, coordinates and weights of integration points (as a consequence of the number of subelements which had to be generated), or the partial sub-volumes of the subelements, above and below the cut.
- **array** `ShapefunctionsAndDerivativesFEM` and `ShapefunctionsAndDerivativesXFEM`: Matrices containing the precomputed standard and enriched element shape functions Φ_i and $\Phi_i\Psi_i$, and their respective derivatives.
- **double** `ElementMassMatrix` and `ElementStiffnessMatrix`: Matrices containing the pre-computed element mass and elemenet stiffness blocks, possibly being updated in every Newmark time step.

As soon as the cut is defined, the data structure is filled in a preprocessing step and can be used for all subsequent calculations. Note that this works for pre-defined cuts only, however, does not hold anymore when dealing with progressive cutting.

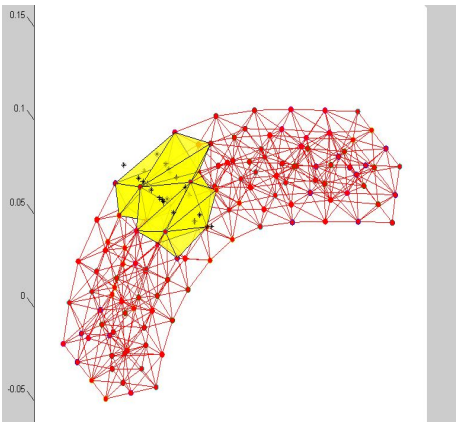
2. The Main Algorithm for the Solution of the Elasticity and Cutting Problem. The main algorithm deals with the actual dynamic elasticity and cutting simulation, which is executed by means of the *implicit Newmark time integration scheme* in order enforce the simulation's stability. In comparison to the classical FEM code, there are only few adaptations to be made for the X-FEM.

As the X-FEM code must consider a *variable number of DOFs per node*, with respect to both, the solution vectors, as well as the system matrices. Therefore, case distinctions must be made for dealing with enriched and non-enriched elements.

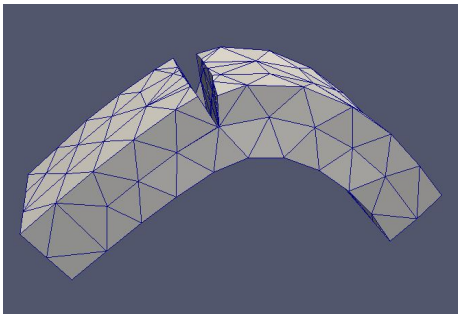
Concerning the *incorporation of boundary conditions* into an X-FEM-based simulation, again, the imposition of Neumann boundary conditions is straightforward and can be realized as in the classical FEM by evaluating corresponding integrals in the weak form along the Neumann boundary. Opposedly, Dirichlet conditions generally require further efforts due to the doubled number of DOFs, compare, e.g., [2]. However, when applying shifted enrichment functions, we can again directly transfer the results from the classical to the extended FEM, since the nodal displacements are directly stored in the standard FEM part of vector $\boldsymbol{u}$ as explained in the previous section, which hence allows for fixing a node to a specific value by simply constraining a single DOF.

Given the fact that cutting strongly affects the deformation of soft tissue, it obviously makes sense to incorporate the *corotation-based formulation* in the X-FEM simulation for the purpose of more accurate results, too. In a dynamic simulation, this involves the *calculation of the rotation matrices* in every time step, since – as we found in the matrix assembly section – the current step's stiffness matrix and its stiffness blocks only depend on the current rotation and on constant force derivatives, the latter of which are suggested to be precomputed. As previously shown, different rotations have to be computed for the subelements above and below the cutting plane. Again, this is performed by extracting the rotation from the deformation gradient tensor $\nabla\boldsymbol{\varphi}$ using polar decomposition. However, we cannot use the original elements' nodal points anymore, but instead must take the vertices of the subelements above and below the cutting plane, and their respective displaced positions as given in equation (42).

3. Postprocessing Steps: Simulation Output and Visualization. In a postprocessing step, and generally when an output is needed for the visualization, the X-FEM requires special adjustments, too. Besides from nodal displacements which – when using shifted enrichment functions – are directly stored in the $\boldsymbol{u}_i$ just like in the standard FEM, there are the contributions of the $\boldsymbol{a}_i$, which account for inner-element displacements and hence the discontinuities.



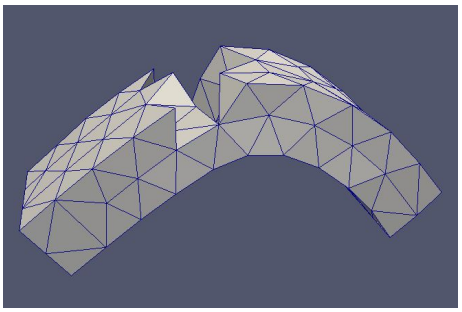
(a) Standard FEM Visualization of a cutting scenario using Matlab.



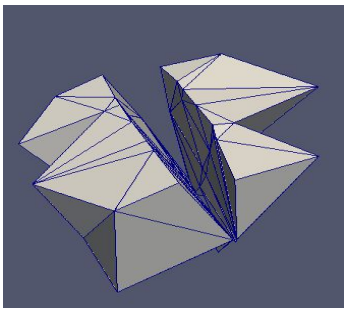
(b) X-FEM Visualization of a cutting scenario using ParaView.

Figure 7: Visualization of a cutting scenario using standard FEM and special X-FEM visualization methods.

Figures 7a and 7b show a cutting scenario applying a standard FEM visualization method (Figure 7a), and a special X-FEM visualization method (Figure 7b). In the standard visualization, both the enriched (yellow) and the non-enriched (white) elements are visualized as uncut tetrahedra defined by their nodal points and continuously connected without any visible gap in between. By contrast, in the adapted X-FEM visualization, the cut becomes visible as the postprocessing considers the contributions of the $\boldsymbol{a}_i$ and hence the inner-element displacements.



(a) First Part of the X-FEM Visualization showing the non-enriched elements only.



(b) Second Part of the X-FEM Visualization showing the subelements of the enriched elements only.

Figure 8: Details of the X-FEM Visualization with respect to the cutting scenario presented in Figure [7].

Figures 8a and 8b show the underlying X-FEM data structure, which differentiates between enriched and non-enriched elements. Non-enriched elements are represented as in the standard FEM, however, enriched elements are represented by their subelements above and below the cut such that a gap opens up.

6 Evaluation

The presented simulation is evaluated by means of a convergence analysis and with respect to stability issues. We define application-relevant simulation test cases and compare our simulation results to the ap-

proved reference results of standard software. Moreover, we consider the impact of applying the linearized and the corotation-based formulations in static and dynamic simulations with respect to the stability.

6.1 Notes on the Evaluation Methods

Reference Solutions. In order to perform the evaluation, we compare our simulation results to approved reference solutions. Using Abaqus[6], we obtain a solution for the linearized calculations when modifying the program's parameters in a way such that the material properties represent the linear, isotropic material of our soft tissue. Since Abaqus does not support the corotational method, we have to apply custom-implemented methods and applications provided as part of the SOFA[7] simulation framework in order to produce reference solutions for the corotation-based simulation. Concerning the cutting simulation, our strategy for enabling comparison and error calculation consists in providing Abaqus and SOFA with input files where the cut is already perfectly remeshed, i.e., where the elements on either side of the cut are disconnected and aligned with the cut's course. In this way, the reference calculation operates on a pre-cut object. Moreover, all calculations performed during the evaluation process are based on the static consideration of final state of the elasticity and cutting simulation, since a time-dependent step by step comparison is not possible for the above mentioned reasons.

Calculation of the Error. This paragraph will define the applied error measure and then outline the error computation algorithm.

We apply the L^2-error, which integrates the difference of the reference solution $\boldsymbol{\varphi}_{\text{ref}}$ and the calculated solution $\boldsymbol{\varphi}_{n_{\text{DOFs}}}$ over the whole domain of the body. The calculated solution $\boldsymbol{\varphi}_{n_{\text{DOFs}}}$ indicates the underlying mesh consisting of $\frac{1}{3}\, n_{\text{DOFs}}$ nodal points. The L^2-error is defined as

$$L^2\text{-error} = \left\|\boldsymbol{\varphi}_{\text{ref}} - \boldsymbol{\varphi}_{n_{\text{DOFs}}}\right\|_2 = \sqrt{\int_\Omega \left(\boldsymbol{\varphi}_{\text{ref}} - \boldsymbol{\varphi}_{n_{\text{DOFs}}}\right)^2 dV} = \sqrt{\sum_{i=1}^{n_{\text{elem}}} \omega_i \left(\left(\boldsymbol{\varphi}_{\text{ref}} - \boldsymbol{\varphi}_{n_{\text{DOFs}}}\right)(\boldsymbol{x}_i)\right)^2}. \quad (83)$$

When using structured meshes, which we do in the evaluation process, the number of DOFs n_{DOFs} directly depends on the characteristic element size h, such that $n_{\text{DOFs}} := n_{\text{DOFs}}(h) = O(h^{-3})$. The same holds for the total number of nodes n_{nodes} and the number of elements n_{elem}.

With the above definition of the error measure, the simulation method converges if

$$\left\|\boldsymbol{\varphi}_{\text{ref}} - \boldsymbol{\varphi}_{n_{\text{DOFs}}}(h)\right\|_2 \overset{h\to 0}{\longrightarrow} 0 \quad (84)$$

where $h \to 0$ if $n_{\text{DOFs}}(h) \to \infty$. It satisfies the order p of convergence if for a constant c

$$\left\|\boldsymbol{\varphi}_{\text{ref}} - \boldsymbol{\varphi}_{n_{\text{DOFs}}}\right\|_2 \leq ch^p\,, \quad (85)$$

holds for $h \to 0$, or $n_{\text{DOFs}}(h) \to \infty$, respectively. Logarithmical transformation under the assumption of equality yields respectively

$$\log\left(\left\|\boldsymbol{\varphi}_{\text{ref}} - \boldsymbol{\varphi}_{n_{\text{DOFs}}}\right\|_2\right) = \log\left(ch^p\right) = \log(c) + p\log(h)\,, \quad (86)$$

$$\log\left(\left\|\boldsymbol{\varphi}_{\text{ref}} - \boldsymbol{\varphi}_{n_{\text{DOFs}}}\right\|_2\right) = \log\left(\tilde{c}n_{\text{DOFs}}^{-\tilde{p}}\right) = \log(\tilde{c}) - \tilde{p}\log(n_{\text{DOFs}})\,, \quad (87)$$

Graphically interpreted, when plotting $\log\left(\left\|\boldsymbol{\varphi}_{\text{ref}} - \boldsymbol{\varphi}_{n_{\text{DOFs}}}\right\|_2\right)$ for different values of respectively h or $n_{\text{DOFs}}(h)$ versus respectively $\log\left(ch^p\right)$ or $\log(\tilde{c}n_{\text{DOFs}}^{-\tilde{p}})$, the slope corresponds to the order p or $\tilde{p}$ of convergence.

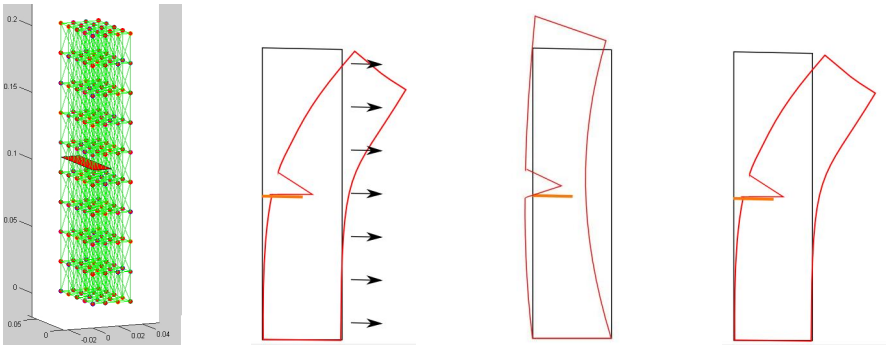
In order to show the convergence of our simulation method, we compare a high resolution reference solution $\boldsymbol{\varphi}_{\text{ref}}$ with our Matlab simulation results $\boldsymbol{\varphi}_{n_{\text{DOFs}}}$ and the corresponding meshes. When increasing the mesh resolution (i.e., $h \to 0$), the error with respect to the reference solution should decrease.

[6] Abaqus is a suite of software applications released by the company Simulia, see http://www.simulia.com/

[7] SOFA (Simulation Open Framework Architecture) primarily targets at real-time simulation, with an emphasis on medical simulation, see http://www.sofa-framework.org/

Test Object and Test Cases. The evaluation builds on three different application-relevant test scenarios which are introduced below. We point out the, corresponding real-world situations in the *medical context*. The underlying test object is an elastic beam as presented in Figure 9a. It consists of homogeneous, almost incompressible, elastic material, with the Poisson ratio $\nu = 0.35$ and the elasticity module $E = 300\,\text{kPa}$. The object's mass is assumed to be uniformly distributed, such that the mass density is $\rho = 1,070.0\,\frac{\text{kg}}{\text{m}^3}$ all over the body. The cutting plane is defined as the set of all points $(x, y, z)^T$ in S_{cut}, where

$$S_{\text{cut}} = \left\{(x, y, z)^T : x \in [0.0, 0.015],\, y \in [0.0, 0.06],\, z = 0.1\right\} .$$



(a) The test object (including the cut) in its initial configuration.

(b) Test case 1: Gravitation.

(c) Test case 2: Stretching.

(d) Test case 3: Bending.

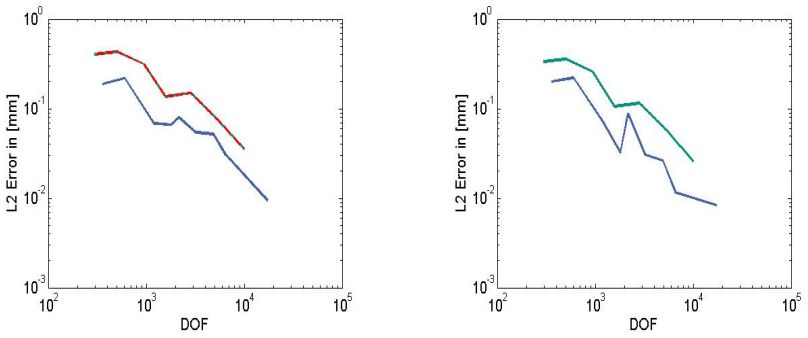
Figure 9: The test object and a schematical depiction of the test cases.

The test cases (compare Figures 9b, 9c, 9d) are defined by different boundary conditions (BCs). We generally assume the lower face of the object (lying in the x-y-plane) to be fixed by means of Dirichlet or displacement BCs for all three test cases. The object's boundary parts which are not subject to a special prescribed deformation are left to natural BCs. In *test case 1*, the object is subject to gravitation forces, where the mass acceleration caused by gravity is $9.81\,\frac{\text{m}}{\text{s}^2}$, acting in positive x-direction. Corresponding situations in the medical context occur whenever a cut is performed and gravity acts on non-fixed semi-cut or sliced parts, pulling them towards the earth. In *test case 2*, the upper left boundary edge $\{(x, y, z)^T : x = 0.0,\, y \in [0.0, 0.06]\}$ to be pulled by 10% in positive z-direction in order to reach a displacement of $z_{\text{deformed}} = z_{\text{initial}} + 0.02$. This causes the whole beam to be stretched. A real-world situation as described by this case might occur, e.g., when in a surgical operation a surgeon wants to open up a gap between the two sides of the cut, hence pulling the soft tissue on the one side of the cut away from the other, in order to be able to glance a view inside the body or to get inside with a surgical instrument. *Test case 3* assumes an additional displacement BC to pull the upper right boundary edge $\{(x, y, z)^T : x = 0.03,\, y \in [0.0, 0.06]\}$ towards a prescribed displacement of $x_{\text{deformed}} = x_{\text{initial}} + 0.03$ and $z_{\text{deformed}} = z_{\text{initial}} - 0.015$, such that the resulting cut beam is bended as it was under the effect of gravity. In the medical context, this case can be found as a combination of the previous two.

As explained above, when testing the linear and corotational versions of the classical FEM in Abaqus and SOFA, we let the cut already be prescribed in the input file. Opposed to this, for testing the X-FEM versions, the cut is performed actively on an arbitrary mesh during the simulation. For both versions, we evaluate the simulated behaviour of the object after the cut.

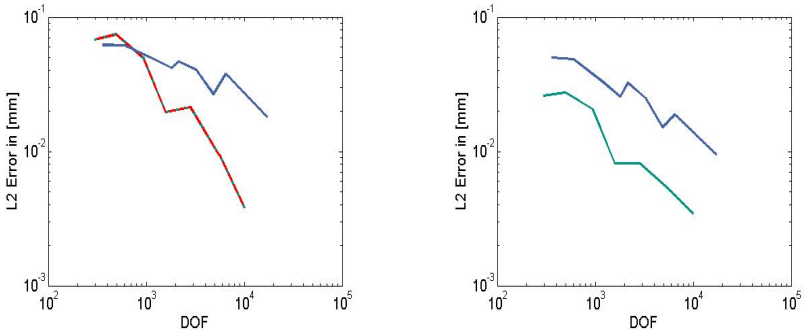
6.2 Evaluation Results

Convergence. We calculate the L^2-error with respect to different mesh resolutions, in terms of the number of DOFs. Figures 10 depict the convergence analysis. We observe the expected overall result: In all three test cases and for both the linear and the corotational versions, the error decreases for increasing mesh resolution.



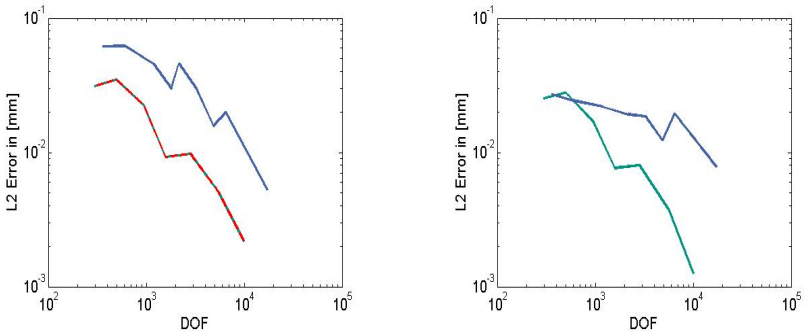
(a) Test case 1, linearized version.

(b) Test case 1, corotated version.



(c) Test case 2, linearized version.

(d) Test case 2, corotated version.



(e) Test case 3, linearized version.

(f) Test case 3, corotated version.

Figure 10: Convergence analysis for the three test cases, for the linearized (left) and corotation-based (right) FEM and X-FEM versions. The plots show the convergence of the simulation results of our methods implemented in Matlab as compared to the reference simulation results of Abaqus and SOFA.

The left side shows the convergence plots for the three test cases for the simulation based on the linearized methods. The grey dotted line represents the calculated errors when comparing Abaqus simulations based on low resolution meshes to the high resolution result, as also calculated in Abaqus. The red dotted line represents the results of our linearized version of the classical FEM compared to the highly resolved reference results of Abaqus. As expected, these two lines completely agree with each other, which means that when given the same mesh our linearized Matlab simulation delivers exactly the same results as the Abaqus simulation. The blue line represents the results when comparing our X-FEM-based simulation to the highly resolved reference solution of Abaqus.

In test case 1, we observe a better convergence of our X-FEM simulation in comparison to the stan-

dard FEM. Opposed to this, for test case 2, the two methods cannot be clearly distinguished in their convergence behavior, and in test case 3, the convergence of the X-FEM simulation is slightly worse than the convergence of the standard FEM.

This obviously is due to the fact that the X-FEM-based simulation as implemented in Matlab does not allow elements to be partially cut. Hence, the cut is continued through the rest of the element, as opposed to the simulation in Abaqus which is based on a mesh that perfectly aligns with the cut. Consequently, the cut in our X-FEM simulation is a little 'longer' than compared to the simulation of Abaqus.

However, this partly contradicts to the observation of a decreasing error for an increasing number of DOFs, i.e., for a decreasing characteristic element size h. We would expect that the higher a mesh is resolved, the more the size of the X-FEM cut approaches the actual size of the cut as prescribed for the Abaqus FEM solution. Thus, the reason for the lower error of the X-FEM solution in test case 1 is to be found in the conditions imposed by the scenario simulated in the test case. This becomes obvious when comparing test cases 1 and 3, which deform in a similar way, however, under the effect of different boundary conditions. In test case 1, gravity causes the deformation. The X-FEM cut, being a little too long, offers a bigger affected target volume to gravity, which consequently shows stronger effects. Opposedly, the prescribed displacement, as given in test case 3, does not benefit from the larger X-FEM cut as compared to the standard FEM cut.

The right side of Figure 10 shows the results of the corotation-based simulations. For all three test cases, we observe that the order of convergence and the general course of the convergence curve approximately correspond to the respective characteristics of the simulations based on the linearized formulation. Generally delivering comparable results, apart from that, the reason for differences can be found in the effects of the absence of ghost forces, as opposed to their presence for large deformations in the linearized versions.

Generally, considering that we created the underlying meshes on the basis of doubling the number of elements in each coordinate direction one after the other, the mesh is not scaled isotropically, which explains the zigzag of the convergence plot.

Note that, due to relatively low resolutions that could be handled by our Matlab simulation, the described convergence analysis has limited significance only and is not conclusive. For more reliable results, simulations on the basis of higher resolved meshes will have to be analyzed.

Stability. The elasticity and cutting simulations based on our dynamic X-FEM implementation are generally very stable. This is due to the novel combination of the underlying corotational formulation and the implicit Newmark time integration scheme. In test case 2 a static corotation-based simulation reveals instabilities such that the object may start jiggling. Georgii focussed on this phenomenon, and figured the reason for the instability to be found in the polar decomposition used for the extraction of the rotation matrices. He therefore alternatively suggests applying a more advanced method based on the determination of minima in the energy potential by means of Newton iterations, see [10]. Yet, since real-world applications require dynamic simulations, these possible instabilities are reduced, oppressed or even eliminated by damping and inertia.

Additional Notes. Besides from the analysis of convergence and stability properties, we again want to mention the enormous potential for optimization in terms of precomputing complex parts before the actual simulation, and the thus improved simulation performance.

6.3 Conclusion

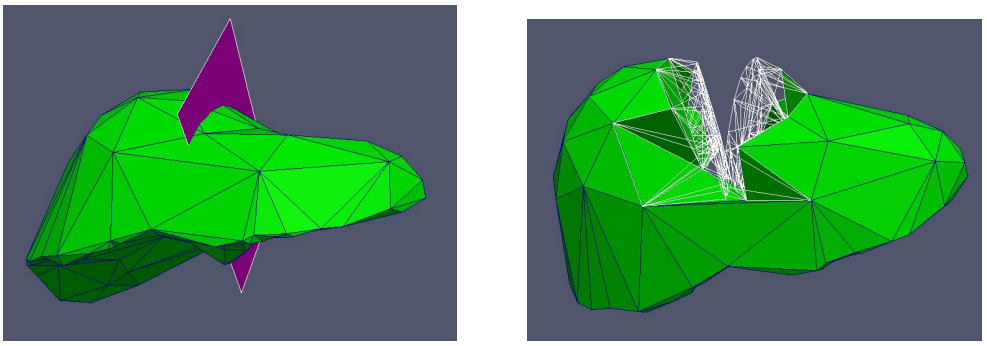
The convergence analysis along with the appendant error plots show the convergence of our X-FEM-based simulation presented in this work with respect to a gold standard reference solution. The X-FEM effectively models discontinuities, enables arbitrary cuts, and shows very good accuracy in terms of DOFs, even in comparison to gold standard FEM simulations which operate on meshes which already perfectly align with the cut before the start of the simulation.

The incorporation of the corotational formulation allows a realistic simulation of large deformations and overcomes the disturbing phenomenon of ghost forces. In addition to that, an important and distinguishing feature of the methods implemented is their potential for precomputing complex parts of the

calculations before the actual time-stepping algorithm. This makes them feasible for real-time applications, such as in the medical and surgical context. To this regard, the application of the implicit Newmark time integration scheme enforces the simulation's stability, which is to be noted as a prominent feature of the implementation, since it represents an important step towards the applicability of the produced simulations in real-world medical simulation systems. Moreover, to the best of the author's knowledge, this is the first X-FEM simulation based on an implementation which combines the corotational formulation with an implicit time integration scheme, hence enforcing almost unconditional stability.

Finally, the particular choice of enrichment functions results in surprisingly sparse stiffness matrices that remain reasonably conditioned as the mesh is refined. This not only allows for an economical use of storage, but also increases computing efficiency by reducing the time-consumption of the simulation.

The flexibility of our X-FEM-based elasticity and cutting simulation in the context of medical engineering is shown in Figure 11, where a cut is performed through an arbitrary unstructured mesh.



(a) A tetrahedralized liver model and the cutting plane.

(b) The result of the cutting simulation. The enriched elements are marked by a white wireframe representation.

Figure 11: Cutting scenario for a human liver: Pre- and post-cutting visualization.

Summarizing, the considerations on the convergence, stability, performance and flexibility of our simulations show their applicability in real-world applications.

Future and ongoing work in this context might address items such as partial and progressive cutting, which so far is not implemented, but for which the X-FEM and the presented data structures offer the right functionalities. Moreover, the extension to multiple cuts is straightforward, since an additional displacement within an element can easily be calculated by twice adding further DOFs according to equation (42) (see [13]). Comparing our elasticity simulation using the standard FEM to our X-FEM implementation, the capabilities of applying higher order isoparametric elements seems promising and would need to be evaluated, too, both with respect to performance and to accuracy.

7 Appendix

7.1 Calculation of the Linearized Version of the Stiffness Matrix for the Elastic Forces

Aiming for maximal exploitation of the computational advantages arising through the simplifications due to the use of the linear elasticity theory, we give a detailed derivation of the calculation and assembly of the linearized element stiffness matrix ${}_{elem}\mathbb{K}^{\text{lin}}$ as in (25).

According to equation (27), we find the linearized $[3n \times 3n]$ element stiffness matrix

$$ {}_{elem}\mathbb{K}^{\text{lin}} = \nabla_{\boldsymbol{\mathcal{U}}} \boldsymbol{F}^{\text{lin}} = \frac{\partial \boldsymbol{\mathcal{F}}^{\text{lin}}}{\partial \boldsymbol{\mathcal{U}}} = \frac{\partial \boldsymbol{\mathcal{F}}^{\text{lin}}}{\partial \tilde{\boldsymbol{\mathcal{P}}}} , \tag{88} $$

where $\boldsymbol{\mathcal{F}}^{\text{lin}}$ implies the strain $\boldsymbol{\varepsilon}$ and the stress $\boldsymbol{\sigma}$ to be replaced by their linearized versions $\boldsymbol{\varepsilon}^{\text{lin}}$ and $\boldsymbol{\sigma}^{\text{lin}}$.

Hence, calculating the integrand of the stiffness term in the equilibrium equation (21) yields

$$\delta\boldsymbol{\varepsilon}^{\text{lin}} \,:\, \boldsymbol{\sigma}^{\text{lin}} = \frac{1}{2}\left(\nabla\delta\boldsymbol{u} + \nabla\delta\boldsymbol{u}^T\right) \,:\, \boldsymbol{\sigma}^{\text{lin}} \overset{\boldsymbol{\mathcal{F}}^{\text{lin}}:=\nabla\Phi\,\boldsymbol{\sigma}^{\text{lin}}}{=} \delta\boldsymbol{\mathcal{U}} \,:\, \boldsymbol{\mathcal{F}}^{\text{lin}}\,, \tag{89}$$

where the linearized element force matrix reads as

$$\boldsymbol{\mathcal{F}}^{\text{lin}} = \left[\sum_{l=1}^{3} \nabla\Phi_{il}\,\boldsymbol{\sigma}_{lp}^{\text{lin}}\right]_{i=1\ldots n,\, p=1\ldots 3}\,. \tag{90}$$

Similarly to (23), for the entries $\boldsymbol{F}_{lj}^{\text{lin}}$ of the global matrices we find

$$\boldsymbol{F}_{lj}^{\text{lin}} = \sum_k \int_{V_k} \boldsymbol{\mathcal{F}}_{ij}^{\text{lin}}\, dV_k\,, \quad \text{where } l = 1\ldots N,\ i = 1\ldots n,\ j = 1\ldots 3\,, \tag{91}$$

i.e., the sum of nodal elastic forces $\boldsymbol{\mathcal{F}}_{ij}^{\text{lin}}$ from all the elements k containing the vertex l in any i-th row.

Finally, as in (27), we find the representation of the Jacobian

$${}_{elem}\mathbb{K}_{ij}^{\text{lin}} = \left(\nabla_{\boldsymbol{\mathcal{U}}}\boldsymbol{F}^{\text{lin}}\right)_{i,j} = \frac{\partial\boldsymbol{\mathcal{F}}_{ip}^{\text{lin}}}{\partial\boldsymbol{\mathcal{U}}_{jq}} = \frac{\partial\boldsymbol{\mathcal{F}}_{ip}^{\text{lin}}}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}}\,. \tag{92}$$

Thus, inserting the expression for the nodal elastic forces, we retrieve

$$\frac{\partial\boldsymbol{\mathcal{F}}_{ip}^{\text{lin}}}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}} = \frac{\partial}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}}\left(\sum_{l=1}^{3}\nabla\Phi_{il}\,\boldsymbol{\sigma}_{lp}^{\text{lin}}\right) = \sum_{l=1}^{3}\frac{\partial}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}}\left(\nabla\Phi_{il}\,\boldsymbol{\sigma}_{lp}^{\text{lin}}\right) = \sum_{l=1}^{3}\nabla\Phi_{il}\,\underbrace{\frac{\partial}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}}\left(\boldsymbol{\sigma}_{lp}^{\text{lin}}\right)}_{\text{term 1}}\,. \tag{93}$$

As a preparing step for the calculation of term 1 we determine the partial derivatives of the linearized strain tensor

$$\begin{aligned}
2\frac{\partial\boldsymbol{\varepsilon}_{ip}^{\text{lin}}}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}} &= \frac{\partial}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}}\left(\nabla\boldsymbol{\varphi}_{ip} + \nabla\boldsymbol{\varphi}_{pi} - 2\boldsymbol{I}\right) = \frac{\partial}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}}\nabla\boldsymbol{\varphi}_{ip} + \frac{\partial}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}}\nabla\boldsymbol{\varphi}_{pi} \\
&\overset{\text{product rule}}{=} \delta_{iq}\left(\nabla\Phi\right)_{jp} + \delta_{qp}\left(\nabla\Phi\right)_{ji} \\
&\overset{\text{symmetry of }\boldsymbol{\varepsilon}^{\text{lin}}}{=} 2\frac{\partial\boldsymbol{\varepsilon}_{pi}^{\text{lin}}}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}}\,,
\end{aligned} \tag{94}$$

such that with the linear material law of Hooke (5) for linear isotropic materials, we obtain the partial derivatives of the linearized stress tensor

$$\begin{aligned}
\frac{\partial\boldsymbol{\sigma}_{ip}^{\text{lin}}}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}} &= 2\mu\frac{\partial\boldsymbol{\varepsilon}_{ip}^{\text{lin}}}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}} + \lambda\delta_{ip}\sum_{k=1}^{3}\frac{\partial\boldsymbol{\varepsilon}_{kk}^{\text{lin}}}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}} \\
&= \mu\left(\delta_{iq}\left(\nabla\Phi\right)_{jp} + \delta_{pq}\left(\nabla\Phi\right)_{ji}\right) + \frac{1}{2}\lambda\delta_{ip}\sum_{k=1}^{3}\underbrace{2\frac{\partial\boldsymbol{\varepsilon}_{kk}^{\text{lin}}}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}}}_{=2\nabla\Phi_{jq}} \\
&= \mu\delta_{iq}\nabla\Phi_{jp} + \mu\delta_{pq}\nabla\Phi_{ji} + \lambda\delta_{ip}\nabla\Phi_{jq}\,.
\end{aligned} \tag{95}$$

Summarizing, i.e., inserting the above results, we obtain the components of the linearized element stiffness matrix ${}_{elem}\mathbb{K}^{\text{lin}}$, i.e., the linearized version of the symmetric force derivatives

$$\begin{aligned}
\frac{\partial\boldsymbol{\mathcal{F}}_{ip}^{\text{lin}}}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{jq}} &= \sum_{l=1}^{3}\nabla\Phi_{il}\,\left(\mu\delta_{lq}\nabla\Phi_{jp} + \mu\delta_{pq}\nabla\Phi_{jl} + \lambda\delta_{lp}\nabla\Phi_{jq}\right) \\
&= \mu\nabla\Phi_{iq}\nabla\Phi_{jp} + \mu\delta_{pq}\sum_{l=1}^{3}\nabla\Phi_{il}\nabla\Phi_{jl} + \lambda\nabla\Phi_{ip}\nabla\Phi_{jq} = \frac{\partial\boldsymbol{\mathcal{F}}_{jq}^{\text{lin}}}{\partial\tilde{\boldsymbol{\mathcal{P}}}_{ip}}\,.
\end{aligned} \tag{96}$$

8 Acknowledgements

This work was carried out with the support of the German Research Foundation (DFG) within the project I03 of the Collaborative Research Center SFB/TRR 125 'Cognition-Guided Surgery'.

References

[1] K.J. Bathe. Finite element method. *Wiley Encyclopedia of Computer Science and Engineering*, 2009.

[2] T. Belytschko, WK Liu, and B. Moran. *Nonlinear finite elements for continua and structures*, volume 36. Wiley, 2000.

[3] T. Belytschko, Y.Y. Lu, and L. Gu. Element-free galerkin methods. *International journal for numerical methods in engineering*, 37(2):229–256, 1994.

[4] T. Belytschko, G. Zi, J. Xu, and J. Chessa. The extended finite element method for arbitrary discontinuities. *Computational Mechanics-Theory And Practice. Barcelona, Spain: CIMNE*, 2003.

[5] D. Bielser, V.A. Maiwald, and M.H. Gross. Interactive cuts through 3-dimensional soft tissue. In *Computer graphics forum*, volume 18, pages 31–38. Wiley Online Library, 1999.

[6] D. Braess. *Finite elemente*. Springer, 2003.

[7] P.G. Ciarlet. Mathematical elasticity: General preface 1. *Studies in Mathematics and its Applications*, 29:v–viii, 2000.

[8] S. Cotin, H. Delingette, and N. Ayache. A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation. *The Visual Computer*, 16(8):437–452, 2000.

[9] Thomas-Peter Fries and Ted Belytschko. The extended and generalized finite element method: An overview of the method and its applications. *International Journal for Numerical Methods in Engineering*, 84(3):253–304, 2010.

[10] J. Georgii and R. Westermann. Corotated finite elements made fast and stable. 2008.

[11] N.J. Higham and R.S. Schreiber. Fast polar decomposition of an arbitrary matrix. 1990.

[12] L. Jerabkova. *Interactive Cutting of Finite Elements based Deformable Objects in Virtual Environments*. Citeseer, 2007.

[13] L. Jerábková and T. Kuhlen. Stable cutting of deformable objects in virtual environments using xfem. *Computer Graphics and Applications, IEEE*, 29(2):61–71, 2009.

[14] A. Mazura. Virtuelles schneiden in volumendaten. 1997.

[15] D.I.J. Mezger. Simulation and animation of deformable bodies. *update*, 2007.

[16] J. Mezger, B. Thomaszewski, S. Pabst, and W. Straßer. Interactive physically-based shape editing. *Computer Aided Geometric Design*, 26(6):680–694, 2009.

[17] H.W. Nienhuys and A.F. van der Stappen. Supporting cuts and finite element deformation in interactive surgery simulation. 2001.

[18] LaraM. Vigneron, JacquesG. Verly, and SimonK. Warfield. Modelling surgical cuts, retractions, and resections via extended finite element method. In Christian Barillot, DavidR. Haynor, and Pierre Hellier, editors, *Medical Image Computing and Computer-Assisted Intervention MICCAI 2004*, volume 3217 of *Lecture Notes in Computer Science*, pages 311–318. Springer Berlin Heidelberg, 2004.

[19] C. Wieners. Numerik partieller differentialgleichungen 2. 2010.

[20] G. Zi and T. Belytschko. New crack-tip elements for xfem and applications to cohesive cracks. *International Journal for Numerical Methods in Engineering*, 57(15):2221–2240, 2003.

[21] O.C. Zienkiewicz and R.L. Taylor. The finite element method (3 volume-set). 2005.

Preprint Series of the Engineering Mathematics and Computing Lab

recent issues

No. 2013-03 Martin Wlotzka, Vincent Heuveline, Edwin Haas, Steffen Klatt, David Kraus, Klaus Butterbach-Bahl, Philipp Kraft, Lutz Breuer: Dynamic Simulation of Land Management Effects on Soil N2O Emissions using a coupled Hydrology-Ecosystem Model on the Landscape Scale

No. 2013-02 Martin Baumann, Jochen Förstner, Jonas Kratzke, Sebastian Ritterbusch, Bernhard Vogel, Heike Vogel: Model-based Visualization of Instationary Geo-Data with Application to Volcano Ash Data

No. 2013-01 Martin Schindewolf, Björn Rocker, Wolfgang Karl, Vincent Heuveline: Evaluation of two Formulations of the Conjugate Gradients Method with Transactional Memory

No. 2012-07 Andreas Helfrich-Schkarbanenko, Vincent Heuveline, Roman Reiner, Sebastian Ritterbusch: Bandwidth-Efficient Parallel Visualization for Mobile Devices

No. 2012-06 Thomas Henn, Vincent Heuveline, Mathias J. Krause, Sebastian Ritterbusch: Aortic Coarctation simulation based on the Lattice Boltzmann method: benchmark results

No. 2012-05 Vincent Heuveline, Eva Ketelaer, Staffan Ronnas, Mareike Schmidtobreick, Martin Wlotzka: Scalability Study of HiFlow^3 based on a Fluid Flow Channel Benchmark

No. 2012-04 Hartwig Anzt, Armen Beglarian, Suren Chilingaryan, Andrew Ferrone, Vincent Heuveline, Andreas Kopmann: A unified Energy Footprint for Simulation Software

No. 2012-03 Vincent Heuveline, Chandramowli Subramanian: The Coffee-table Book of Pseudospectra

No. 2012-02 Dominik P.J. Barz, Hendryk Bockelmann, Vincent Heuveline: Electrokinetic optimization of a micromixer for lab-on-chip applications

No. 2012-01 Sven Janko, Björn Rocker, Martin Schindewolf, Vincent Heuveline, Wolfgang Karl: Software Transactional Memory, OpenMP and Pthread implementations of the Conjugate Gradients Method - a Preliminary Evaluation

No. 2011-17 Hartwig Anzt, Jack Dongarra, Vincent Heuveline, Piotr Luszczek: GPU-Accelerated Asynchronous Error Correction for Mixed Precision Iterative Refinement

No. 2011-16 Vincent Heuveline, Sebastian Ritterbusch, Staffan Ronnås: Augmented Reality for Urban Simulation Visualization

No. 2011-15 Hartwig Anzt, Jack Dongarra, Mark Gates, Stanimire Tomov: Block-asynchronous multigrid smoothers for GPU-accelerated systems

No. 2011-14 Hartwig Anzt, Jack Dongarra, Vincent Heuveline, Stanimire Tomov: A Block-Asynchronous Relaxation Method for Graphics Processing Units

No. 2011-13 Vincent Heuveline, Wolfgang Karl, Fabian Nowak, Mareike Schmidtobreick, Florian Wilhelm: Employing a High-Level Language for Porting Numerical Applications to Reconfigurable Hardware

Preprint Series of the Engineering Mathematics and Computing Lab (EMCL)

www.emcl.iwr.uni-heidelberg.de