ENGINEERING MATHEMATICS
AND COMPUTING LAB

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Simulation of Complex Cuts in Soft Tissue with the Extended Finite Element Method (X-FEM)

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

Preprint No. 2014-02

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. 2014-02

Affiliation of the Authors

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

[a] *Project-team Simulation in Healthcare using Computer Research Advances (SHACRA) – INRIA Lille – North Europe Research Center*
[b] *Institute for Anthropomatics, Karlsruhe Institute of Technology (KIT), Germany*
[c] *Engineering Mathematics and Computing Lab (EMCL), Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Germany*
[1] *Corresponding Author: Christoph J. Paulus, christophjpaulus@gmail.com*

Impressum

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

Speyerer Str. 6,
69115 Heidelberg
Germany

Simulation of Complex Cuts in Soft Tissue with the Extended Finite Element Method (X-FEM)

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

December 16, 2014

Abstract

In this paper we present an approach to model discontinuities in the solution of the elasticity problem without changing the initial grid topology. In the context of surgery simulation or real-time intraoperative registration this method allows for adapting a finite element model during the operation in the presence of cutting or resection.

We outline a formulation of the eXtended finite element method (X-FEM) for elastic solids and present an approach for modeling arbitrary cuts by means of finite elements on tetrahedral grids. For this purpose, completely cut elements are enriched with the shifted Heaviside function and partially cut elements are enriched with so-called asymptotic crack tip functions. In this context we show how to handle the geometry of partial cuts and how the necessary local coordinate system based on polar coordinates is constructed. Finally, we present a flexible implementation of the approach.

A numerical validation shows that the approach can handle complex cuts through low resolution geometries. Furthermore, a convergence analysis reveals that the approach is superior to standard remeshing techniques in terms of accuracy per degrees of freedom. The source code of the presented method is available under an open-source license[1].

1 Introduction

In this paper we make use elasticity theory in order to model cuts through deformable models. We apply the eXtended finite element method (X-FEM) and discretize the problem using an oversampling integration method. We outline the challenges we are facing and explain the advantages that arise from an implementation of this method on top of an existing finite element method (FEM) framework. Finally, we evaluate the simulation and point out possible improvements.

The chapter on the continuous formulation of a cutting problem starts with an introduction to elasticity theory. We formulate the cutting problem in the differential – strong – and in the variational – weak – form. The weak formulation will be used in the third chapter where the discretization of the cutting problem is outlined. In this context, the necessary adaptations of the object mesh are explained, such that the mesh of the object can be opened at its intersections with the mesh of the cut. After discretization and adjustment of the grid of the object, we discretize the function space by defining the shape functions that are the base for the classic or the continuous Galerkin finite element method (CG FEM), proceeding with the declaration of the shape functions for the X-FEM. Replacing the continuous function in the variational formulation of chapter2 by an interpolated discrete function yields an integral equation

[1] Code hosting platform bitbucket - https://bitbucket.org.
Download command: git clone https://chrijopa@bitbucket.org/chrijopa/xfem-in-cpp.git

which can be transformed into an equation integrating over the body's set of tetrahedra. Calculating the integral over a tetrahedron by means of the oversampling method finally yields the discretized cutting problem.
Related to this work, we implemented the X-FEM in C++ to solve and simulate the cutting problem. In chapter 4 we describe the geometric precomputations and our implementation.
For the evaluation of our method we adapted object meshes to a cut such that we can obtain solutions of a cutting problem with the CG FEM. We compared these solutions to our results applying the root mean square error – an approximation of the Lebesgue error. We obtain very good results and our method even surpasses the solutions of the CG FEM specifically adapted to the cutting problem. Further details and conclusions to the evaluation can be found in chapter 5.
We conclude by summarizing this work and by pointing out possible future directions.

The contributions of this work are:

- application of the X-FEM to the simulation of cutting in soft tissue – with almost arbitrary cuts
- capability of simulating completely and partially cut elements
- an evaluation that demonstrates the superior accuracy in comparison with CG FEM
- very precise and application-related description of the X-FEM.

This preprint is intended to help researchers learn more about the X-FEM by working on it hands on. Please note that the corresponding source code and further documentation – the evaluation and examples – is available under an open source license. Please do not hesitate to contact the authors if you have any questions on the code.

1.1 Related work

Simulating surgical cuts with finite element based discretization techniques is a challenging problem. A straight forward approach is to simply remove the elements along the cutline [4] [5]. However, this procedure does not only lead to a poor approximation of the cut, but it also violates the conservation of mass. In order to obtain a more accurate solution, local re-meshing around the cut has to be performed [13]. This approach can easily lead to ill-shaped tetrahedra (so called *sliver* elements) that can impair not only the performance, but also the stability of the simulation. Although complex procedures have been proposed that address this problem [2] [17], the main drawbacks of the re-meshing approach still remain; the computationally demanding mesh adaptation procedure, the generation of many additional degrees of freedom as well as potential stability issues.

Novel methods thus seek to model the discontinuity in the solution without changing the grid topology. One possibility is to enrich the elements along the cutting front with additional, discontinuous basis functions. This approach is called the eXtended finite element method (X-FEM) [8]. Vigneron et al. use this method in conjunction with a linear elastic model to compensate the *brain shift* that occurs during craniectomy in the presence of resected tissue [18] [19]. A first application of an X-FEM based technique for real-time cutting 3D deformable objects was presented by Jerabkova et al. [11]. The work by Hegemann et al. showed how X-FEM based techniques can be used simulate complex fracture phenomena in the context of computer animation [9]. Kaufmann et al. showed how the approach can be used to model very complex cuts through shell elements [12].

A different approach to the accurate simulation of surgical cutting is to use a hierarchical, high resolution hexahedral grid [10] [7]. The idea of this method is to simply model the cut by disconnecting the grid along the element boundaries, by deleting the so called links between the elements. This simplifies not only the geometrical handling and the implementation of the cutting problem, but also enables the simulation of multiple cuts. Furthermore, the solution of the arising linear system of equations can be easily accelerated using multigrid solvers due to the regular grid hierarchy. However, very high resolution grids have to be used in order to achieve a decent accuracy. Thus, the method needs much more degrees of freedom in comparison with X-FEM based techniques in order to achieve the same accuracy. One possibility to reduce the number of degrees of freedom is to group elements together in order to form so called composite elements [20]. However, this procedure reduces the accuracy for corotated elasticity and fully non-linear formulations.

2 Continuous formulation of the cutting problem

2.1 Introduction to elasticity theory

In elasticity theory the body is seen as a continuum with reversible deformation, which is a deterministic approach. The *deformation* $\boldsymbol{\varphi}$ can be formulated using the *displacement function* $\boldsymbol{u}$:

$$\boldsymbol{\varphi} : \begin{cases} \mathbb{R}^3 \supset \Omega & \longrightarrow \tilde{\Omega} \subset \mathbb{R}^3 \\ \boldsymbol{x} & \mapsto \tilde{\boldsymbol{x}} = \mathbf{id}(\boldsymbol{x}) + \boldsymbol{u}(\boldsymbol{x}) \end{cases}$$

We use a small strain approximation, and apply the *linearized Cauchy elasticity tensor*

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right).$$

We obtain a linear dependency between tension and distortion because of the linearization. We suppose that the body behaves the same in all directions and at all points, i.e., we have an isotropic and homogeneous material. Assuming that the body is not subject to tension in the initial configuration, yields the following simplification of the *stress tensor*:

$$\sigma_{ij} = 2\mu\varepsilon_{ij} + \delta_{ij}\lambda\sum_{k=1}^{3}\varepsilon_{kk}$$

2.2 Elasticity problem - differential formulation

We consider object deformations enforced by boundary conditions like displacements $\boldsymbol{u}_D$ on Γ_D – *Dirichlet boundary conditions* – and surface forces $\boldsymbol{s}$ on Γ_N (where $\Gamma_N \cap \Gamma_D = \emptyset$) – *Neumann boundary conditions*. Then we can formulate all boundary conditions

$$\begin{aligned} \boldsymbol{u}_D : \Gamma_D &\longrightarrow \mathbb{R}^3 \quad \text{and} \\ \boldsymbol{s} : \Gamma_N &\longrightarrow \mathbb{R}^3. \end{aligned}$$

In addition to the boundary condition, the object's *body forces* are given by $\boldsymbol{b} : \Omega \longrightarrow \mathbb{R}^3$.

One can obtain the *differential* or *strong formulation* of the *elasticity problem* by applying the force equilibrium on infinitesimal small volumes of the object, see [1]:

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

$$\begin{aligned} -\mathrm{div}(\boldsymbol{\sigma}(\boldsymbol{x})) &= \boldsymbol{b}(\boldsymbol{x}) && \forall \boldsymbol{x} \in \Omega \\ \boldsymbol{\sigma}(\boldsymbol{x})\boldsymbol{n}(\boldsymbol{x}) &= \boldsymbol{s}(\boldsymbol{x}) && \forall \boldsymbol{x} \in \Gamma_N \\ \boldsymbol{u}(\boldsymbol{x}) &= \boldsymbol{u}_D(\boldsymbol{x}) && \forall \boldsymbol{x} \in \Gamma_D \\ \partial\Omega &= \Gamma_N \cup \Gamma_D && \Gamma_D \cap \Gamma_N = \emptyset \end{aligned}$$

where $\boldsymbol{n}$ is the normal of Ω pointing outwards. Defining the spaces $V_B(\Omega, \boldsymbol{f}) = \{\boldsymbol{u} | -\mathrm{div}(\boldsymbol{\sigma})_{|\Omega} = \boldsymbol{f}\}$ $V_N(\Gamma_N, \boldsymbol{f}) = \{\boldsymbol{u} | (\boldsymbol{\sigma} \cdot \boldsymbol{n})_{|\Gamma_N} = \boldsymbol{f}\}$ and $V_D(\Gamma_D, \boldsymbol{f}) = \{\boldsymbol{u} | \boldsymbol{u}_{|\Gamma_D} = \boldsymbol{f}\}$ for an arbitrary function $\boldsymbol{f}$ helps to simplify the strong formulation, such that we obtain

Find $\boldsymbol{u} = \boldsymbol{\varphi} - \mathbf{id} \in \mathbb{C}^2(\Omega) \cap \mathbb{C}^1(\overline{\Omega}) \cap V_B(\Omega, \boldsymbol{b}) \cap V_N(\Gamma_N, \boldsymbol{s}) \cap V_D(\Gamma_D, \boldsymbol{u}_D)$

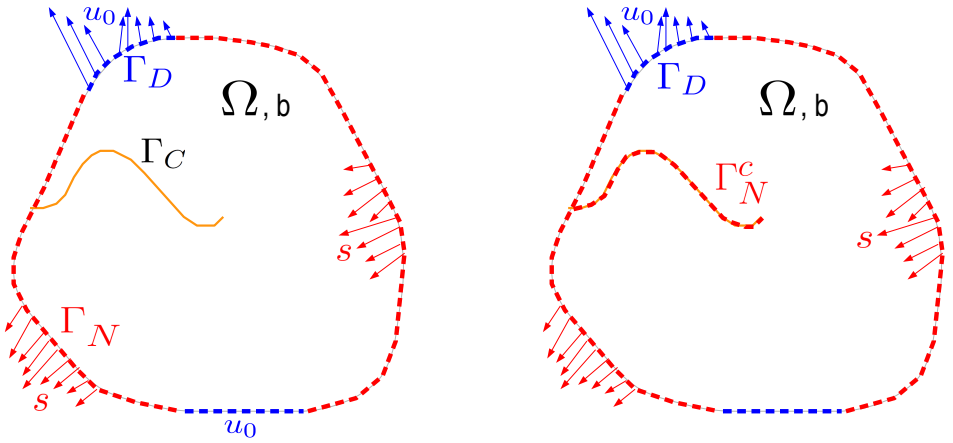
.

2.3 Cutting problem - differential formulation

A realistic cutting simulation must allow for an unrestrictive definition of an arbitrary cut. We firstly define the cut[2] surface $\Gamma^c \subset \mathbb{R}^3$. For simplicity reasons we only consider one cut, i.e. the parameterized cut boundary is continuous. However, the approach can be extended in order to accomodate multiple cut configurations. Since the cut separates the body, the newly considered domain is $\Omega^c = \Omega \backslash \Gamma^c$. As a result of that, the boundary $\partial\Omega^c$ of the body is larger: $\partial\Omega^c = \partial\Omega \cup (\Gamma^c \cap \Omega)$. On the additional boundary we neither allow the application of a force boundary condition which is unequal zero nor an application of a displacement boundary condition, so we have $\boldsymbol{s}_{|\Gamma^c \cap \Omega} = \mathbf{0}$.
Defining $\Gamma^c_N = \Gamma_N \cup \Gamma^c$, the cutting problem corresponds to an elasticity problem, and we obtain the following *strong formulation of the cutting problem*:

$$\boxed{\text{Find } \boldsymbol{u} = \boldsymbol{\varphi} - \mathbf{id} \in \mathbb{C}^2(\Omega^c) \cap \mathbb{C}^1(\overline{\Omega^c}) \cap V_B(\Omega^c, \boldsymbol{b}) \cap V_N(\Gamma^c_N, \boldsymbol{s}) \cap V_D(\Gamma_D, \boldsymbol{u}_D)}$$

In figure 1 we can see an example of a cutting problem in 2D.



(a) Elasticity problem with a cut

(b) Interpretation as a cutting problem with extended boundary force condition

Figure 1: 2D example of a cutting problem with force and displacement boundary conditions

2.4 Weak formulation

Multiplying the above differential equation with an arbitrary test function $\delta\boldsymbol{u} \in \mathbb{C}^\infty_0(\Omega) \cap V_D(\Gamma_D, \mathbf{0})$ and integrating the equation over the space Ω^c, we obtain an integral equation that is equivalent to the differential equation. Using the *Gauß integration theorem*, this integral formulation can be transformed to the so-called *variational* or *weak formulation* of the problem

$$\boxed{\begin{array}{c}\text{Find } \boldsymbol{u} = \boldsymbol{\varphi} - \mathbf{id} \in H^1(\Omega) \cap V_D(\Gamma_D, \boldsymbol{u}_D) \cap V_N(\Gamma^c_N, \boldsymbol{s}), \text{ such that} \\ \int_\Omega (\nabla\delta\boldsymbol{u})_{ij}\sigma_{ji}\, d\Omega = \int_{\Gamma_N} \delta\boldsymbol{u}^T \cdot \boldsymbol{s}\, d\Gamma_N + \int_\Omega \delta\boldsymbol{u}^T \cdot \boldsymbol{b}\, d\Omega \qquad \forall\delta\boldsymbol{u} \in \mathbb{C}^\infty_0(\Omega) \cap V_D(\Gamma_D, \mathbf{0}) \quad (1)\end{array}}$$

where $H^1(\Omega)$ is the *Sobolev space* over Ω. In this formulation we apply $a_i b_i = \sum_i a_i b_i$, if there are doubled indices in a multiplication. This also holds for the rest of this work.

[2] cut related variables use a superscript c, for example v^c

3 Discretization

The solution of the problem formulated in chapter 2 can only be solved numerically if we discretize the problem. We begin by discretizing the spatial domain, i.e., the object and the cut. We proceed by adjusting the mesh of the object to the mesh of the cut. Then we declare the discretized function spaces. Together with the numerical integration this serves as the basis of the linear equation system that needs to be solved in order to obtain a numerical solution of the problem.

3.1 Discretization of the object

In the following declarations, N_A denotes the number of elements in an arbitrary set $A = \{a_i\}$, i.e. $A = \{a_i | 0 \leq i < N_A\}$. To solve the elasticity problem numerically, we first discretize the object using a finite number of *points* $\boldsymbol{P}_k \in \overline{\Omega}, k < N_{\mathcal{P}}$ and form a *set of points* $\mathcal{P} = \{\boldsymbol{P}_k | k < N_{\mathcal{P}}\} \subset \overline{\Omega}$. We connect the points $\boldsymbol{P}_k \in \mathcal{P}$ by tetrahedra and write the global node ids for each tetrahedron in the connectivity vector *connect* $C = (c_m), m \in \{0, ..., 3\}$. Then each tetrahedron T is the convex hull of its four nodes, i.e. $T = \text{conv}(\boldsymbol{P}_{c_0}, \boldsymbol{P}_{c_1}, \boldsymbol{P}_{c_2}, \boldsymbol{P}_{c_3})$. The *set* $\mathcal{T}$ *of* $N_{\mathcal{T}}$ *tetrahedra* T then defines the *discretized object* $\Omega_h = (\bigcup_{T \in \mathcal{T}} T)^0$. The grid Ω_h approximates and converges to Ω for decreasing element sizes $h = \max_{T \in \mathcal{T}}(\text{diam}(T))$.

The goal of the discretization is the calculation of the discrete *deformations* $\boldsymbol{U}_k$ at the points $\boldsymbol{P}_k$. This will be explained in chapters 3.4, 3.5 and 3.6.

3.2 Discretization of the cut

If we aim on solving the cutting problem, the cut needs to be discretized as well. We proceed similarly to chapter 3.1:

We replace the continuous formulation of the cut using a finite number of points $\boldsymbol{P}_k \in \overline{\Gamma^c}^{2D} := \Gamma^c \cup \partial^{2D}\Gamma^c, k < N_{\mathcal{P}^c}$ and form a set of points $\mathcal{P}^c = \{\boldsymbol{P}^c_k | k < N_{\mathcal{P}^c}\} \subset \overline{\Gamma^c}^{2D}$. Here, ∂^{2D} is the boundary of a 2D object in the 3D space, i.e. $\partial^{2D}\Gamma^c$ is a curve that can be parametrized with one variable. Then we connect $\boldsymbol{P}^c_k \in \mathcal{P}^c$ by triangles, each triangle $T^c = \text{conv}(\boldsymbol{P}^c_{c^c_0}, \boldsymbol{P}^c_{c^c_1}, \boldsymbol{P}^c_{c^c_2})$ is the convex hull of its three nodes with the ids $C^c = (c^c_m), m \in \{0, 1, 2\}$. The *set* $\mathcal{T}^c$ *of* $N_{\mathcal{T}^c}$ *triangles* T^c then defines our *discretized cut* $\Gamma^c = \text{int}^{2D}(\bigcup_{T^c \in \mathcal{T}} T^c)$, where int^{2D} denotes the two-dimensional interior.

Furthermore, we define the *normals of the cut triangles* T^c as $\boldsymbol{n}^c = (\boldsymbol{P}^c_{c^c_1} - \boldsymbol{P}^c_{c^c_0}) \times (\boldsymbol{P}^c_{c^c_2} - \boldsymbol{P}^c_{c^c_0})$. We identify normals of two adjacent cut triangles *pointing into the same direction*, if and only if their scalarproduct is greater than zero. We assume that the normals of all cut triangles point into the same direction, i.e. we can not handle a cut that has a shape of a *Möbius strip*. The direction of the normals defines the parts of the object *above* $\Omega_+ \subset \Omega$ and *below* $\Omega_- \subset \Omega$ *the cut* ($\Omega = \Omega_+ \cup \Omega_- \cup \Gamma^c$). Finally the *cut front* can be defined as $\partial^{2D}\Gamma^c \cap \Omega$.

3.3 Adjusting the grid of the object

In this chapter we introduce additional degrees of freedom for the object to account for the mesh of the cut. For the visualization of the cut opening, the intersections of the object mesh and the cut mesh are necessary as well This will be covered in chapter 4.

To distinguish partially and completely cut elements, we start checking whether the edges of an element are cut. An *edge of an element is cut*, if the intersection of the edge and the 2D closure of the cut Γ^c is not the empty set. A *face is completely cut*, if there are two edges on that face, that are cut. Finally, the *element is*

1. *uncut*, if there are no cut edges on that element
2. *partially cut*, if the element has cut edges but less than three completely cut element faces
3. *completely cut* through, if there are three or more completely cut element faces.

The global nodes are called *branch enriched*, if they belong to a partially cut element and *sign enriched*, if they belong to a completely, but not to a partially cut element. Nodes that are neither sign nor branch enriched are called *non-enriched*. At this point, it can be seen that we allow for elements that have two differents kinds of nodes, the so called *blending elements* (an example for blending elements will be provided at the end of this subsection).
The separation of the two parts of a completely cut element is easier to display, than the opening of a partially cut element. Therefore the complexity of completely cut elements is lower than the complexity of partially cut elements. Since the branch enriched nodes belong to the partially cut elements, they need to be able to store a lot more information. Therefore each of those nodes will get an additional enrichment of twelve degrees of freedom, in contrast to the sign enriched nodes, that get an additional enrichment of three degrees of freedom. The additional enrichments correspond to additional displacements $\boldsymbol{U}_k \in \mathbb{R}^3$, i.e., we obtain four additional displacements at each branch enriched and one additional displacement at each sign enriched node. We enumerate the displacements such that the ids $k_\mathcal{P}$ of the displacement of the initial nodes, the ids k_S of the displacements corresponding to the sign enrichment and the ids k_B of the displacements corresponding to the branch enriched nodes hold the condition $k_\mathcal{P} < k_S < k_B$.
In the following, N_S is the number of sign enriched and N_B the number of branch enriched nodes. Then the number of displacements $\boldsymbol{U}_k$ in our adjusted model is $N = N_\mathcal{P} + N_S + 4N_B$ and we have $k_\mathcal{P} \in \{0, \dots, N_\mathcal{P} - 1\}, k_S \in \{N_\mathcal{P}, \dots, N_\mathcal{P} + N_S - 1\}$ and $k_B \in \{N_\mathcal{P} + N_S, \dots, N - 1\}$.

For the next steps, we define in each element the *local node id* $l_m \in L$ for every global id $c_m \in C, m \in \{0, \dots, 3\}$, which is in all current elements $L = (l_0, l_1, l_2, l_3)^T = (0, 1, 2, 3)^T$. For each enriched node the connect C of a completely or partially cut element T is extended with the global number k of the displacement $\boldsymbol{U}_k$ and the local connect L is extended with the local node number. Additionally, a branch enriched node of an uncut element extends the connect C with the global number of the displacement that corresponds to the branch enrichment and the local connect L with the local node number. That means that uncut elements which are neighbours of partially cut elements can be blending elements, since they have standard nodes and branch enriched nodes. The variable L will be used in chapter 3.4: the discretization of the function space.

The example displayed in figure 2 illustrates the explained functionality in two dimensions.

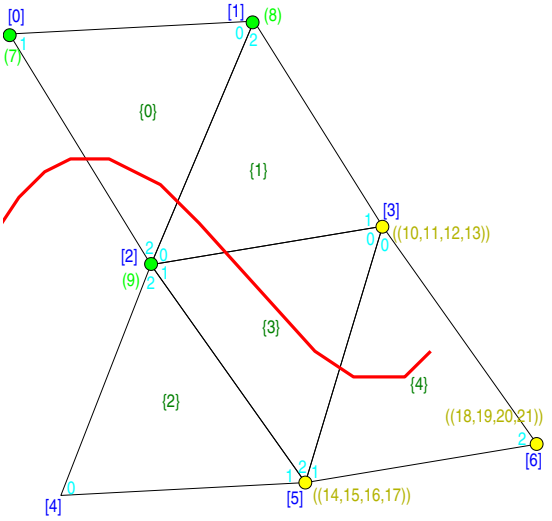


Figure 2: Example for the enumeration of the displacements of the initial nodes in [blue], the local node ids in cyan, the ids k of the displacements corresponding to sign enrichment in (green) and branch enrichment in ((yellow)). And finally the elements ids in {dark green}

In the example, the number of nodes for the standard mesh is $N_{\mathcal{P}} = 7$ and the number of enriched nodes is $N_S = 3, N_B = 3$, so we obtain the total number of displacements $N = 22$. The variables that are dependent on the element number are the following:

0. $C = (1, 0, 2, 8, 7, 9)^T, L = (0, 1, 2, 0, 1, 2)^T$
1. $C = (2, 3, 1, 8, 9, 10, 11, 12, 13)^T, L = (0, 1, 2, 2, 0, 1, 1, 1, 1)^T$
2. $C = (4, 5, 2, 14, 15, 16, 17)^T, L = (0, 1, 2, 1, 1, 1, 1)^T$
3. $C = (3, 2, 5, 9, 10, 11, 12, 13, 14, 15, 16, 17)^T, L = (0, 1, 2, 1, 0, 0, 0, 0, 2, 2, 2, 2)^T$
4. $C = (3, 5, 6, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21)^T$,
   $L = (0, 1, 2, 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2)^T$

In this case the elements 1 and 3 are completely cut blending elements and the element 2 is an uncut blending element. As it can be seen in the example, we are ordering the connect of the elements in such a way, that the displacement of the initial nodes appears first, afterwards the displacement that corresponds to sign and branch enriched nodes.

3.4 Discretizing the function space - defining the shape functions

The displacement function $\boldsymbol{u}$ used in chapter 2 depends on an infinite number of points. In the following, the displacement function will be approximated by functions that are only dependent on a finite number of degrees of freedom. In case there is no cut, the classical finite element method, the so-called continuous Galerkin finite element method (CG FEM), can be used. We begin with the definition of the shape functions for the CG FEM, i.e., for the displacements $\boldsymbol{U}_k$ with the ids $k < N_{\mathcal{P}}$ and proceed with the shape functions for the extended finite element method (X-FEM), i.e., for displacements $\boldsymbol{U}_k$ with ids $N_{\mathcal{P}} \leq k < N$.

We start with defining a *reference tetrahedron* $\hat{T} = \text{conv}(\hat{\boldsymbol{P}}_0, \hat{\boldsymbol{P}}_1, \hat{\boldsymbol{P}}_2, \hat{\boldsymbol{P}}_3)$ where $\hat{\boldsymbol{P}}_0 = (1, 0, 0)^T, \hat{\boldsymbol{P}}_1 = (0, 1, 0)^T, \hat{\boldsymbol{P}}_2 = (0, 0, 0)^T, \hat{\boldsymbol{P}}_3 = (0, 0, 1)^T$, and the *natural coordinates*

$$\hat{N}_0(\hat{\boldsymbol{x}}) = \hat{x}_0, \qquad \hat{N}_1(\hat{\boldsymbol{x}}) = \hat{x}_1, \qquad \hat{N}_2(\hat{\boldsymbol{x}}) = 1 - \hat{x}_0 - \hat{x}_1 - \hat{x}_2, \qquad \hat{N}_3(\hat{\boldsymbol{x}}) = \hat{x}_2.$$

Every point $\hat{\boldsymbol{x}} \in \hat{T}$ can be mapped into an arbitrary tetrahedron T, by using the *element transformation*

$$\begin{aligned} \boldsymbol{\zeta} : \hat{T} \longrightarrow T, \qquad & \boldsymbol{\zeta}(\hat{\boldsymbol{x}}) = \boldsymbol{P}_{c_m} \hat{N}_m(\hat{\boldsymbol{x}}) = \boldsymbol{x}, \\ \boldsymbol{\zeta}^{-1} : T \longrightarrow \hat{T}, \qquad & \boldsymbol{\zeta}^{-1}(\boldsymbol{x}) = \text{inv}(\boldsymbol{\zeta})(\boldsymbol{x}) = \hat{\boldsymbol{x}}. \end{aligned}$$

For the definition of the shape functions, we use an arbitrary tetrahedron $T \in \mathcal{T}$. The connect C is sorted, such that we find the ids of the initial nodes in the entries $m \in \{0, \ldots, 3\}$. Only the displacements of those nodes are considered in the CG FEM. We use linear shape functions. We define the *isoparametric shape functions* for each tetrahedron T in the *nodal formulation* Φ_{c_m} and in the *elementary formulation* N_m:

$$\Phi_{c_m|T}(\boldsymbol{x}) = N_m(\boldsymbol{x}) = \hat{N}_{l_m}(\boldsymbol{\zeta}^{-1}(\boldsymbol{x})), \qquad \boldsymbol{x} \in T$$

At this point and in the following formulations of the shape functions, we assume that the elementary functions N_m are only defined on their tetrahedron, i.e., $N_m : T \to \mathbb{R}$ and the support of the nodal shape functions Φ_k is restricted to the elements that have a displacement $\boldsymbol{U}_k$ with the id k, i.e. $k \in C$. The shape functions in the elementary formulation describe the discretized displacement of an uncut element that has no partially cut neighbour:

$$\boldsymbol{u}_{h|T}(\boldsymbol{x}) = \boldsymbol{U}_{c_m} N_m(\boldsymbol{x})$$

The nodal shape functions define the piecewise linear displacement function $\boldsymbol{u}_h$ that would be used in the CG FEM, i.e., in a problem without a cut:

$$\boldsymbol{u}_h(\boldsymbol{x}) = \sum_{k=0}^{N_{\mathcal{P}}-1} \boldsymbol{U}_k \Phi_k(\boldsymbol{x})$$

The nodal shape functions of the conventional finite element method are linear on each element, satisfy the partition of unity and the Kronecker delta property.

Now, we define the additional shape functions that arise because of the cut, i.e. we look at a tetrahedron T with $N_C > 4$. We start with the shape functions for additional displacements corresponding to sign enrichment. Therefore we choose a tetrahedron T with an $m < N_C : c_m \in \{N_\mathcal{P}, \dots, N_\mathcal{P} + N_S - 1\}$. Then the nodal shape function Φ_{c_m} and the elementary shape function N_m are defined as

$$\begin{aligned} \Phi_{c_m|T}(\boldsymbol{x}) = N_m(\boldsymbol{x}) = \hat{N}_{l_m}(\boldsymbol{\zeta}^{-1}(\boldsymbol{x}))\psi_{l_m}(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in T, \\ \text{where } \psi_m(\boldsymbol{x}) = \frac{H(\boldsymbol{x}) - H_m}{2} \quad \text{with} \\ H(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{x} \in \Omega_+ \\ -1, & \boldsymbol{x} \in \Omega_- \end{cases}, \text{ and } H_m = H(\boldsymbol{P}_{c_m}). \end{aligned} \tag{2}$$

We call $H(\boldsymbol{x})$ the *Heaviside function* and $\psi_m(\boldsymbol{x})$ the *shifted Heaviside function*. For more information on the choice of the enrichment function ψ_m, I am referring to Schoch et al [15].

As previously mentioned, the branch enriched nodes are part of a tetrahedron, that may have a complex opening since they are connected to an element which is partially cut. Therefore the formulation of the shape functions for displacements that correspond to a branch enrichment is more difficult than the formulation of the shape functions for displacements corresponding to sign enrichment. Like the shape functions of displacements that originate in a sign enrichment, the natural coordinates $\hat{N}_m$ will be multiplied with *enrichment functions*. For the displacements corresponding to branch enrichment we choose the *asymptotic crack tip functions* (ACTFs) as enrichment functions. The ACTFs are dependent on the polar coordinates (r, θ) that correspond to the closest cut front. In order to obtain the polar coordinates, we will first transform $\boldsymbol{x}$ to $\boldsymbol{x}_{\boldsymbol{abc}}$, i.e., to a coordinate system $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ that is adjusted to the closest cut front and afterwards to the polar coordinates (see figure 3). After that we can define the ACTFs and the

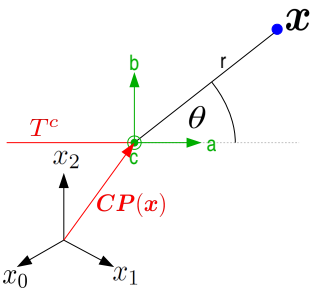


Figure 3: The orientation of the different coordinate systems with respect to each other.

shape functions for the branch enriched degrees of freedom.
The ids k of branch enriched nodes satisfy $N_\mathcal{P} + N_S \leq k < N$, so we choose a tetrahedron T with an $m < N_C : c_m \in \{N_\mathcal{P} + N_S, \dots, N - 1\}$ and an arbitrary $\boldsymbol{x} \in T$. We start with calculating the point $\boldsymbol{CP}(\boldsymbol{x})$ of the cut front that is the closest to $\boldsymbol{x}$, i.e.,

$$\boldsymbol{CP}(\boldsymbol{x}) = \underset{\boldsymbol{x}^c \in (\partial^{2D}\Gamma^c) \cap \Omega}{\operatorname{argmin}} \|\boldsymbol{x} - \boldsymbol{x}^c\|$$

Then we proceed by finding the triangle $T^c \in \mathcal{T}^c$ of the discretized cut such that $\boldsymbol{CP}(\boldsymbol{x}) \in T^c$, and we define the cut front line $CF(\boldsymbol{x}) = \text{conv}(\tilde{\boldsymbol{P}}^c{}_0, \tilde{\boldsymbol{P}}^c{}_1)$ with $\tilde{\boldsymbol{P}}^c{}_0, \tilde{\boldsymbol{P}}^c{}_1 \in \{\boldsymbol{P}^c_{c_0}, \boldsymbol{P}^c_{c_1}, \boldsymbol{P}^c_{c_2}\}$ and $\boldsymbol{x}^c \in CF(\boldsymbol{x})$. Then we adjust the $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ coordinate system to the cut triangle T^c with its cut front line $CF(\boldsymbol{x})$ such that

$$\begin{aligned} \boldsymbol{b} &= \|\boldsymbol{n}^c\|, \\ \boldsymbol{c} &= \frac{\tilde{\boldsymbol{P}}^c{}_0 - \tilde{\boldsymbol{P}}^c{}_1}{\left\|\tilde{\boldsymbol{P}}^c{}_0 - \tilde{\boldsymbol{P}}^c{}_1\right\|}, \\ \boldsymbol{a} &= \boldsymbol{b} \times \boldsymbol{c}, \end{aligned}$$

which helps to build the rotation matrix $R = (\boldsymbol{a}|\boldsymbol{b}|\boldsymbol{c})$ and we obtain the transformation of $\boldsymbol{x}$ to the $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ coordinate system

$$\boldsymbol{x}_{\boldsymbol{abc}} = R^{-1}(\boldsymbol{x} - \tilde{\boldsymbol{P}}^c{}_0)$$

and finally the transformation to the 2D (r, θ) polar coordinate system

$$\{r = \sqrt{x_{\boldsymbol{a}}^2 + x_{\boldsymbol{b}}^2}, \qquad \theta = \arctan\left(\frac{x_{\boldsymbol{b}}}{x_{\boldsymbol{a}}} + (\operatorname{sgn}(x_{\boldsymbol{b}}) + 1_{x_{\boldsymbol{a}}>0, x_{\boldsymbol{b}}=0}(x_{\boldsymbol{a}}, x_{\boldsymbol{b}})) \cdot \pi\right)\}.$$

The *asymptotic crack tip function* (ACTF) number can be determined by $n = (c_m - N_{\mathcal{P}} - N_S) \bmod 4$, which helps to define the shape functions

$$\Phi_{c_m|T}(\boldsymbol{x}) = N_m(\boldsymbol{x}) = F_n(r, \theta)\hat{N}_{l_m}(\boldsymbol{\zeta}^{-1}(\boldsymbol{x}))$$

with the ACTFs

$$F(r, \theta) = \left(\sqrt{r}\sin\left(\frac{\theta}{2}\right), \sqrt{r}\cos\left(\frac{\theta}{2}\right), \sqrt{r}\sin\left(\frac{\theta}{2}\right)\sin(\theta), \sqrt{r}\cos\left(\frac{\theta}{2}\right)\sin(\theta)\right)^T.$$

Again, we have that $supp(N_m) = T$ and we can formulate the elementary displacements as:

$$\boldsymbol{u}_{h|T}(\boldsymbol{x}) = \boldsymbol{U}_{c_m} N_m(\boldsymbol{x}).$$

The elementary displacement function can be expanded to the *global displacement function* using the *elementary formulation* by defining $\boldsymbol{u}_h(\boldsymbol{x}) = \boldsymbol{u}_{h|T}(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in T$, i.e.,

$$\boldsymbol{u}_h(\boldsymbol{x}) = \sum_{T \in \mathcal{T}} \boldsymbol{U}_{c_m} N_m(\boldsymbol{x}). \tag{3}$$

The *global displacement function* can be formulated by using the *nodal formulation* of the shape function as well, we obtain

$$\boldsymbol{u}_h(\boldsymbol{x}) = \sum_{k=0}^{N-1} \boldsymbol{U}_k \phi_k(\boldsymbol{x}). \tag{4}$$

Despite the similarities in the formulation of the displacement function, in general the shape functions of the X-FEM are not linear on each element. Furthermore, they do not hold the partition of unity and Kronecker delta property.

With the definitions of the shape functions above, we assure that the discretized displacement function $\boldsymbol{u}_h(\boldsymbol{x})$ approximate the displacement function $\boldsymbol{u}(\boldsymbol{x})$, i.e., $\boldsymbol{\varphi}_h(\boldsymbol{x}) \longrightarrow \boldsymbol{\varphi}(\boldsymbol{x}), h \to 0$. Therefore we can approximate the test functions $\nabla\delta\boldsymbol{u}$ with the elementary formulation $\sum_{T\in\mathcal{T}} (\delta\boldsymbol{U}_{c_m}) \nabla N_m(x)$. Concluding we can merge the theories above: we replace the displacement function $\boldsymbol{u}(\boldsymbol{x})$ and the test functions $\nabla\delta\boldsymbol{u}$ in the weak formulation (1) by their discrete substitute. To simplify the following derivation, we write $\boldsymbol{u}$ and $\boldsymbol{\varphi}$ instead of $\boldsymbol{u}_h$ and $\boldsymbol{\varphi}_h$ until the end of chapter 3.

3.5 Local stiffness matrices

We begin by defining the *element force matrix* $\mathcal{F}_{mi} := (\nabla N_m)_l \sigma_{li}$ for each tetrahedron T. For the following steps, we write U_{ki} for the i-th entry of the displacement vector $\boldsymbol{U}_k$. Then we can use the elementary formulation $(\nabla\delta\boldsymbol{u})_j = \sum_{T\in\mathcal{T}} \delta\boldsymbol{U}_{c_m} (\nabla N_m)_j$ (see formula (3)) and $supp(N_m) = T$ to transform the left hand side of the weak formulation (see formula (1)) to

$$\int_\Omega (\nabla\delta\boldsymbol{u})_{ij}\sigma_{ji} d\Omega = \sum_{T\in\mathcal{T}} \left(\delta U_{c_m i} \int_T \mathcal{F}_{mi} dT\right).$$

The *local stiffness matrix* $\mathbb{k}$ of the tetrahedron T is defined as

$$\begin{aligned} \mathbb{k}_{3n+j,3m+i} &= \int_T \frac{\partial \mathcal{F}_{mi}}{\partial U_{c_n j}} dT \\ &= \int_T \mu\nabla N_{ni}\nabla N_{mj} + \lambda\nabla N_{nj}\nabla N_{mi} + \delta_{ji}\mu\nabla N_{mo}\nabla N_{no}\, dT \end{aligned} \tag{5}$$

with the derivations of the elementary shape functions N_m of the tetrahedron T. In the linear elasticity theory we have a linear dependency between the element force matrix and the local stiffness matrix $\Bbbk$. Using the assumption of no tension in the initial state, i.e., $\mathcal{F}_{mi|U_{kn}=0} = 0$, then yields

$$\int_\Omega (\nabla \delta \boldsymbol{u})_{ij} \sigma_{ji} d\Omega = \sum_{T \in \mathcal{T}} \delta U_{c_m i} \cdot \Bbbk_{3n+j,3m+i} \cdot U_{c_n j}. \tag{6}$$

Defining the *body force vector*

$$\boldsymbol{B}_k = \int_\Omega \Phi_k \boldsymbol{b} \; d\Omega,$$

and the *surface force vector*

$$\boldsymbol{S}_k = \int_{\Gamma_N} \Phi_k \boldsymbol{s} \; d\Gamma_N,$$

we can rewrite the right hand side of (1) replacing $\delta \boldsymbol{u}^T$ by the nodal formulation $\sum_{k=0}^{N-1} (\delta \boldsymbol{U}_k)^T \Phi_k$ (see formula (4)):

$$\int_\Omega \delta \boldsymbol{u}^T \boldsymbol{b} \; d\Omega + \int_\Omega \delta \boldsymbol{u}^T \boldsymbol{s} \; d\Omega = \sum_{k=0}^{N-1} \delta \boldsymbol{U}_k^T \left(\boldsymbol{B}_k + \boldsymbol{S}_k \right). \tag{7}$$

In order to obtain the body force vector $\boldsymbol{B}_k$, the surface force vector $\boldsymbol{S}_k$ and the element stiffness matrices $\Bbbk$, we need to be able to integrate the corresponding functions, which we will deal with in the following subsection.

3.6 Numerical integration

The conventional finite element method is based on polynomial shape functions. In order to calculate the body force vector $\boldsymbol{B}_k$, the surface force vector $\boldsymbol{S}_k$, and the element stiffness matrices $\Bbbk$, those functions need to be integrated over simple domains, like tetrahedra. Most implementations of the CG FEM apply the fast and stable *Gauß integration*. The Gauß integration is advantageous for the CG FEM, since an exact numerical integration of polynomials with low degrees over domains like a tetrahedron can be achieved. The ACTFs are trigonometric functions wherefore the Gauß integration can only approximate the solution of their integrals. Furthermore, the discontinuity of the Heaviside function yields to inaccuracy, since an integration over arbitrary three dimensional domains is necessary. In order to overcome the negative side affects, we expand the idea of the Gauß integration to replacing the integral by a sum of function values multiplied with integration weights.

The integration points in the tetrahedron T are obtained by transforming the *reference integration points* $\chi = (\boldsymbol{\xi}_l)$ where $\boldsymbol{\xi}_l \in \hat{T}, l < N_\chi$. For that, we apply the transformation function $\boldsymbol{\zeta}$ introduced in the last chapter and weigh the function values with the *integration weights* w_l:

$$\begin{aligned}
\int_T \boldsymbol{f} \; dT &= \int_{\boldsymbol{\zeta}^{-1}(T)=\hat{T}} |\det(\nabla \boldsymbol{\zeta}(\hat{\boldsymbol{x}}))| \boldsymbol{f}(\boldsymbol{\zeta}(\hat{\boldsymbol{x}})) d\hat{T} \\
&= \int_0^1 \int_0^{1-x_3} \int_0^{1-x_2-x_3} |\det(\nabla \boldsymbol{\zeta}(\hat{\boldsymbol{x}}))| \boldsymbol{f} \left(\boldsymbol{\zeta}(\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}) \right) dx_1 dx_2 dx_3 \\
&= \left(\sum_{k=0}^{N_\chi - 1} w_k \boldsymbol{f}(\boldsymbol{\zeta}(\boldsymbol{\xi}_k)) |\det(\nabla \boldsymbol{\zeta}(\boldsymbol{\xi}_k))| \right) + \text{Error}(\boldsymbol{f} \circ \boldsymbol{\zeta}, N_\chi)
\end{aligned}$$

We obtain the reference integration points χ by subdividing the tetrahedron into smaller tetrahedra in which we place Gauß integration points. With the numerical integration we obtain the integrals mentioned above. Now we proceed similarly to the CG FEM: we lexicographically write the element stiffness matrix $\Bbbk$ of (5) into a *global stiffness matrix* $\mathbb{K} \in \mathbb{R}^{3N \times 3N}$:

$$\mathbb{K}_{3k_1+j_1,3k_2+j_2} = \sum_{\exists T \in \mathcal{T}: m_1, m_2 \leq N_C \wedge k_1 = c_{m_1} \wedge k_2 = c_{m_2}} \Bbbk_{3m_1+j_1,3m_2+j_2}, \qquad j_1, j_2 \in \{1,2,3\}. \tag{8}$$

With the unknown displacement $\boldsymbol{U}_k, k \in \{0, \ldots, N-1\}$ and exploiting the arbitrary choice of $\delta\boldsymbol{U}_k$ we receive the discretized formulation of the problem using (6), (7) and (8):

Solve $\mathbb{K}_{3k+i,3l+j}U_{lj} = B_{ki} + S_{ki}, \forall k < N, i < 3$, where $u(\boldsymbol{x}) = \sum_{k=0}^{N-1} \boldsymbol{U}_k \Phi_k(\boldsymbol{x}) \in V_D(\Gamma_D \cap \mathcal{P}, \boldsymbol{u}_D)$.

This formulation finally discretizes the continuous cutting problem of chapter 2. In the following chapter, we give more details about the implementation of the X-FEM using C++.

4 Implementation

The aim of this work was to enable a simulation of the surgical cutting caused opening of soft tissue, considered both from the mathematical point of view and also from the implementation point of view. This chapter explains our approach to an implementation of the above described X-FEM-based simulation for the cutting problem.

The focus lies on the implementation of the X-FEM, which is why we directly assume the object and cut to be given and represented as a mesh. An implementation of a dynamic simulation is possible, since the dynamic formulation of the cutting problem does not differ considerably from the static formulation presented in the previous chapters, for further information see [15]. In this context, we assume that the cut can only change the topology of the object at the beginning of the simulation.

Based on those assumptions, the geometrical and topological changes of the mesh (see chapter 3.3) can be precomputed. The first section is dedicated to these geometric precomputations that introduce the topological changes of the mesh and serve as the fundament for the shape functions of the additional degrees of freedom. The second section gives a brief overview of the structure of our implementation.

4.1 Geometric precomputations

In the context of an implementation of the X-FEM, there are two main issues, where geometric functions need to be applied: For the adjustment of the grid and for the calculation of the shape functions and their derivatives.

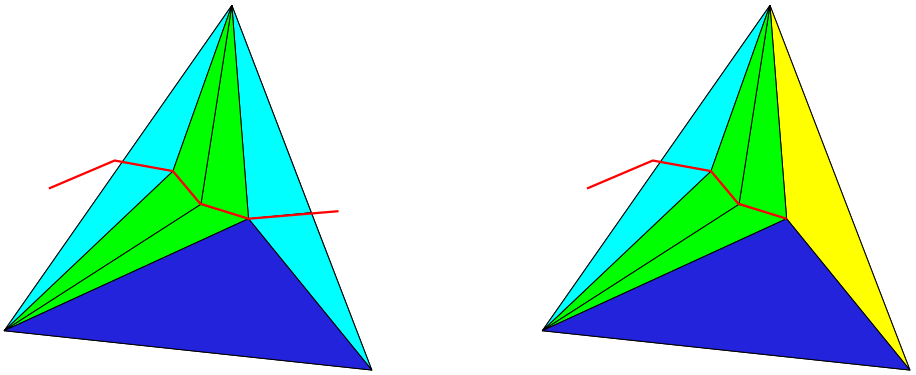
The theory of the adjustments of the grid for the X-FEM was mentioned in chapter 3.3. The application reveals the necessity for several other steps outlined in this paragraph.
We begin by formulating the input in two new ways: we connect the nodes $\boldsymbol{P}_k$ by triangles (the faces of the tetrahedra $T \in \mathcal{T}$) or by lines (the edges of the tetrahedra $T \in \mathcal{T}$). Then, we check whether an edge of the object intersects with the triangular mesh of the cut. Based on this information, the elements are named *uncut*, *completely* or *partially cut*. Concluding, we store the branch and sign enriched nodes and the extended connectivity data of the global node IDs C and local node IDs L of the elements, following the idea of chapter 3.3.
In addition to the adjustment of the grid for the computation, we adapt the grid for a visually realistic output of the calculated results. In order to obtain a visually pleasant result we calculate all the intersections of the mesh of the cut and the mesh of the object and we visualize the nodes according to their real positions by means of the $\boldsymbol{U}_k$ calculated using the X-FEM:

$$\boldsymbol{u}_h(\boldsymbol{x}) = \sum_{k=0}^{N-1} \boldsymbol{U}_k \phi_k(\boldsymbol{x}).$$

Moreover, these newly introduced intersection points are triangulated in order to obtain a suitable visualization mesh. For this purpose, we calculate the intersections between the

- edges of the object and the triangles of the cut, and the
- triangles of the object and the edges of the cut

using a functionality of the toolkit CGAL[3]. Furthermore, we obtain the points $\boldsymbol{P}_k^c \in \mathcal{P}^c$ of the cut that are in the a tetrahedron of the object exploiting that the normals of the faces of the tetrahedron point outwards. Then we connect the intersection points and the cut points in the object by triangles such that we obtain a visually pleasant result.



(a) completely cut face (b) partially cut face

Figure 4: Triangulation of element faces.

Now, we proceed with geometric precomputations for the shape functions of the enriched nodes. The corresponding theory was explained in chapter 3.4.
In order to compute the Heaviside function $H(\boldsymbol{x})$ of $\boldsymbol{x} \in \Omega$, as defined in formula (2), we calculate the cut triangle T^c that is closest to $\boldsymbol{x}$. We compare the vector $\boldsymbol{x} - \boldsymbol{P}_{c_0}^c$ with the normal of the cut triangle $\boldsymbol{n}^c$ and check whether $\boldsymbol{x}$ lies above or below the cut. Based on that information, we precompute the shifted Heaviside function ψ_m of the nodes $\boldsymbol{P}_k \in \mathcal{P}$ and of the integration points $\boldsymbol{\zeta}(\boldsymbol{\xi}_l)$ for each element.
For the shape functions that originate in the branch enrichment, we need to transform the given coordinate system (x_0, x_1, x_2) into a new coordinate system (r, θ). We follow the idea that was described in chapter 3.4 and precompute cut triangles $T^c \in \Gamma^c$ that are part of the cuts' two-dimensional boundary. These cut triangles help to build the rotation matrix R and are the basis of the transformation between the two coordinate systems.

4.2 Implementation structure

Since we introduce some specific declarations and properties in this work in order to allow a discussion about the applied X-FEM model and simulation, the appendant implementation has to overcome the challenge of, on the one hand, sticking to a preferably very general FEM code, and on the other hand, allowing for all additional object properties and features used.

In order to facilitate this requirement, a topology-based 'cut model' class structure is introduced and can – with slight adaptations – be used as an add-on to standard FE implementations. In this chapter, we therefore give a brief overview of the implementation of the X-FEM simulation and the underlying X-FEM cut model code structure. Moreover, we show which external toolkits and software frameworks and libraries it is based on, and give hints on what the implemented functionalities do, and how the input/output interface works, e.g., with respect to other software frameworks, such as the Simulation Open Framework Architecture (SOFA) (see http://www.sofa-framework.org/) or HiFlow[3] (see http://www.hiflow3.org/). Concluding, we depict a stand-alone demonstrator which exemplarily makes use of the most common functionalities.

[3]Computational Geometry Algorithms Library (CGAL): http://www.cgal.org/

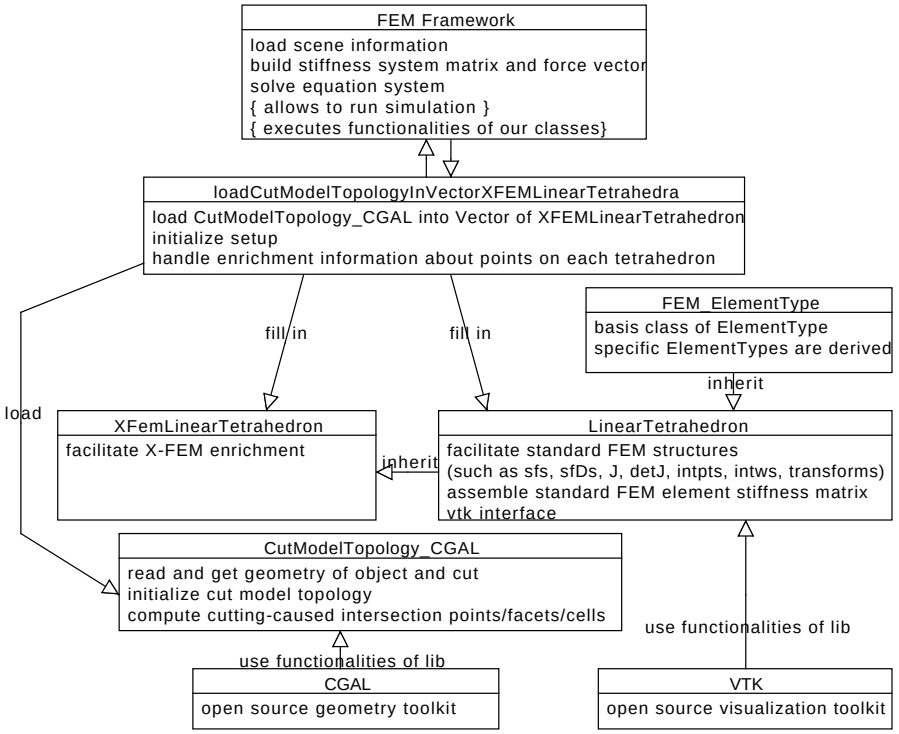


Figure 5: Class structure of X-FEM Simulation.

As can be seen in the unified modeling language (UML) diagram of figure 5, we build the base class 'CutModelTopologyCGAL' on top of CGAL functionalities. In the context of this simulation, the open source framework CGAL is mainly used for geometric operations, i.e., for the steps mentioned in the first subchapter. The CGAL-based class 'CutModelTopologyCGAL' facilitates the handling of all operations on enriched elements, and hence connects a FEM framework with CGAL features without an overlapping of their functionalities, eventually allowing for the exchange of the two above named tools. The basic tasks executed through class 'CutModelTopologyCGAL' are the following:

- load the object's volume mesh (e.g. tetrahedra) and the cut surface (e.g. triangles), and transform these into segments and triangles,
- check the object's mesh nodes on their location above or below the triangulated cut surface,
- check the way elements are cut (i.e., uncut, partially cut, or completely cut), and based on this information determine the type of enrichment (i.e., non-enriched, sign-enriched, or branch-enriched),
- adjust the element connectivity information with respect to the additionally enriched nodes,
- if a (partial or complete) cut goes through an element of the object: calculate the intersections of the object's tetrahedral mesh and the cut's triangle mesh, and obtain the resulting intersection points, as well as in case of partial cuts, obtain the cut boundary which is needed for the asymptotic crack tip functions,
- triangulate the new elements' surfaces.

In order to demonstrate the features of this newly implemented cut model topology, we chose to exemplify our simulation by means of a tetrahedralized mesh Thus, let class 'FEM ElementType' be the basis for the class 'LinearTetrahedron', which is only an option besides e.g. 'QuadraticTetrahedron'. The class 'LinearTetrahedron' provides all functions and variables that are necessary to handle a linear tetrahedron, e.g., the FE shape function and their derivatives, coordinate transformations, integration points and weights, as well as the assembly of the system's stiffness matrix, and the capabilities to perform

displacements of the element. The class uses the Visualization Toolkit (VTK)[4] and its associated mesh formats for file input/output. Similarly, the class 'XFemLinearTetrahedron' inherits all properties of 'LinearTetrahedron', and only adds the X-FEM specific features needed in order to handle enriched elements: the enriched elements' shape functions, their derivatives and the corresponding degrees of freedom (DoFs).

Building on these FE-typical facilitating tasks, the class 'loadCutModelTopologyIntoVectorXFEMLinearTetrahedra' now loads the cut model topology, which is used to initialize the corresponding cut, mesh and DoF characteristics, and transfers the thus obtained information into a vector of the class 'XFEMLinearTetrahedron'. The thus obtained above mentioned element stiffness matrices $\mathbb{k}$ can then be given to a standard Finite Element framework. The Finite Element framework builds the system stiffness matrix $\mathbb{K}$ and computes the solution vector $\boldsymbol{U}_k, k < N$, using Neumann and Dirichlet boundary conditions. In order to give access to the X-FEM simulation functionalities, we implemented a small FEM structure that uses the X-FEM classes mentioned above. The code is published open source on bitbucket[5]. For an example workflow, see figure 6.

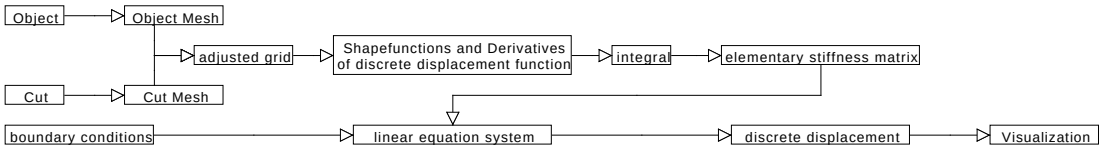


Figure 6: Scheme of X-FEM Simulation workflow.

We point out some restrictive features of the X-FEM and hence our code in the following. Firstly, the body forces $\boldsymbol{b}$ (and surface forces $\boldsymbol{s}$) are assumed to be constant on the elements (and the surfaces, respectively). An extension to nonlinear body and surface forces is straight forward, since the implemented integration method of chapter 3.6 can deal with arbitrary functions. Secondly, as previously mentioned the code applies a zero Neumann boundary condition on the cut Γ_C, which is why the additional entries of the boundary force vector are all zero, i.e. $\boldsymbol{S}_n = \boldsymbol{0}, \quad \forall n > 3N_{\mathcal{P}}$. Thirdly, the body force $\boldsymbol{b}$ has no effect on the enriched nodes, and therefore the additional entries of the body force vector are zero as well, i.e. $\boldsymbol{B}_n = \boldsymbol{0}, \quad \forall n > N_{\mathcal{P}}$. As a consequence the calculation of the body force vector $\boldsymbol{B}_n$ can be executed by the finite element framework and does not have to be provided by our classes.

5 Evaluation

In this chapter we evaluate our implementation of the X-FEM-based simulation. We begin by introducing the tools we apply; the test case, the reference solution and the error measure. We compare the solutions of our algorithm with the solution of the well established CG FEM and interpret the results.

5.1 Test problem, reference solution and error measure

We consider the deformation of an elastic beam with an initial configuration $\Omega = [0, 0.03]m \times [0, 0.06]m \times [0, 0.2]m$ (in the following we will omit the unit m). The material behaviour of the beam is isotropic and homogeneous with a *Young's modulus* of $E = 300kPa$ and a *Poisson's ratio* of $\nu = 0.35$. We fix the beam by a Dirichlet boundary condition at its stump in all directions

$$\boldsymbol{u}_D(x) = \boldsymbol{0} \quad \forall \boldsymbol{x} \in \Gamma_{D_1} = [0, 0.03] \times [0, 0.06] \times \{0\}$$

[4]Visualization Toolkit (VTK): http://www.vtk.org/

[5]Code hosting platform bitbucket - https://bitbucket.org.

Download command: git clone https://chrijopa@bitbucket.org/chrijopa/xfem-in-cpp.git

and stretch the beam with a second Dirichlet boundary condition

$$\boldsymbol{u}_D(x) = \begin{pmatrix} 0 \\ 0 \\ 0.02 \end{pmatrix} \quad \forall \boldsymbol{x} \in \Gamma_{D_2} = \{0\} \times [0, 0.06] \times \{0.2\}$$

The rest of the boundary is in its equilibrim, which is equivalent to applying a Neumann boundary condition equals zero: $\boldsymbol{s}(\boldsymbol{x}) = \boldsymbol{0}$ for $\boldsymbol{x} \in \Gamma_N = \partial\Omega \backslash (\Gamma_{D_1} \cup \Gamma_{D_2})$. We cut the beam halfway at $x_2 = 0.1$ in the x_0-direction, i.e. we have the cut surface

$$\Gamma^c = [0, 0.015] \times [0, 0.06] \times \{0.1\},$$

so the zero Neumann boundary condition expands to $\Gamma_N^c = \partial\Omega \backslash (\Gamma_{D_1} \cup \Gamma_{D_2}) \cup \Gamma^c$.

We compare our results for the cutting problem with a *reference solution*, that is obtained using the CG FEM. With the verified convergence of CG FEM, the reference solution can be applied to investigate on the convergence of our results.

In order to be able to solve the cutting problem with the CG FEM we need to adapt our mesh to the topological changes that arise from the cut. We begin by aligning the surfaces of the elements to the cut. Then the nodes on the cut are doubled, one of the nodes is attached to the elements above the cut and the other one is attached to the elements below the cut. After this adjustment, the object allows for an opening of the cut, since the doubled nodes can move away from each other. The perfect alignment on and separation at the cut, simplifies the cutting problem to a standard elasticity problem. Since the conventional finite element method is sufficient to solve this problem, we can use CG FEM to obtain our reference solution.
To minize the difference between the deformation of the reference solution and the real world, we choose a body with 466560 tetrahedra and 256188 degrees of freedom. These degrees of freedom correspond to the points $\boldsymbol{P}_k^r \in \overline{\Omega}, k < N_{\mathcal{P}^r} = 85396$ and the set of points $\mathcal{P}^r = \{\boldsymbol{P}_k^r | k < N_{\mathcal{P}^r}\}$ discretizes the domain Ω. The CG FEM provides the deformed object, which we will refer to using its deformation function $\boldsymbol{\varphi}^r$.

Our solutions to the cutting problem are stored in the vtk-format. We visualize the solutions using Paraview[6], the reference solution can be seen in figure 7.

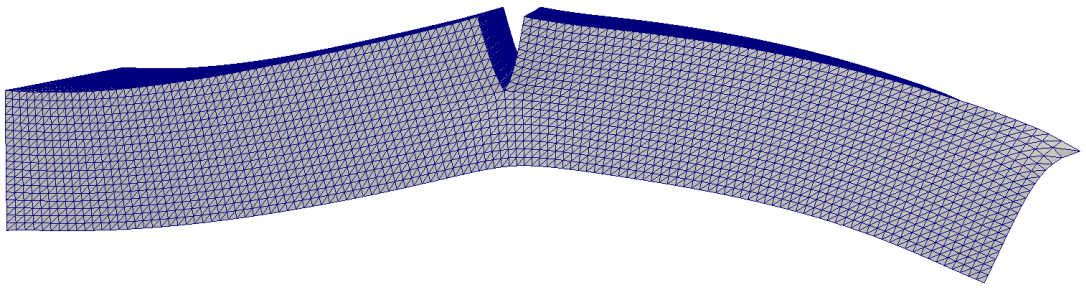


Figure 7: The reference solution

[6] http://www.paraview.org/

Now, we proceed with the examination of the functionality of our method, comparing the reference solution with our results. In the evaluation of the conventional finite element method, a common definition of the deformation error is

$$\epsilon_{L^2} = \|\boldsymbol{\varphi}^r - \boldsymbol{\varphi}_h\|_{L^2(\Omega)} = \sqrt{\int_\Omega (\boldsymbol{\varphi}^r - \boldsymbol{\varphi}_h)^2 d\Omega}.$$

Utilizing this error measure is difficult in our case, since the shape functions of the branch enriched elements only allow for an appoximation of the integral. Moreover, this is computationally expensive, because of the complexity of these shape functions. Therefore, we chose to approximate the error measure ϵ_{L^2} directly instead of approximating the integral. The *Root-Mean-Square Error* (RMS)

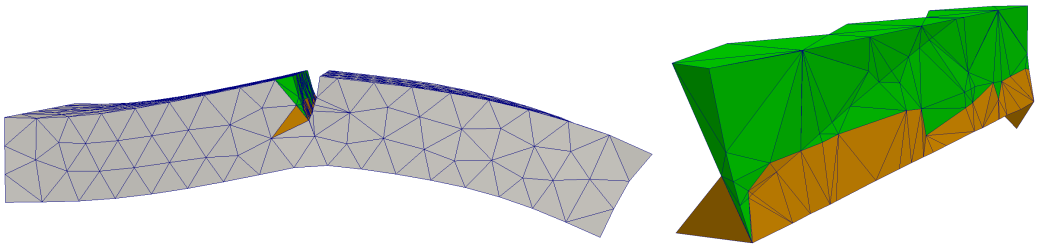
$$\epsilon_{RMS} = \sqrt{\frac{1}{3N_{\mathcal{P}^r}} \sum_{k=0}^{N_{\mathcal{P}^r}-1} \|(\boldsymbol{\varphi}^r - \boldsymbol{\varphi}_h)(\boldsymbol{P}_k^r)\|_2^2},$$

which depends on the nodes $\boldsymbol{P}_k^r$ of the reference solution approximates ϵ_{L^2} very well, since the elements of the reference solution all have the same size. Therefore, we choose the RMS error. We calculate the RMS error, by first calculating the position – i.e., the id of the tetrahedron and the local coordinates in a mesh with lower resolution – of each reference node $\boldsymbol{P}_k^r \in \Omega_h$ and then deform the node by using the displacement function in the elementary formulation (3).

In the course of our examination, we utilize our implementation of the X-FEM in C++ to compute the results of the cutting problem mentioned above. We discretize the beam Ω by means of an arbitrary unstructured grid, that our algorithm can cut at any point. The cut is discretized by two triangles.

The goal of the evaluation was to not only prove the convergence of our implementation of the X-FEM, but also to compare its result. For comparison, we are using the same idea, that has been applied to obtain the reference solution: We introduce meshes with different number of equidistant points, such that the surface of the tetrahedra aligns with the cut. On the cut, we double the points and attach one of them to the tetrahedra above and one of them to the tetrahedra below, allowing an opening of the cut. Following these steps, we simplified the cutting problem to the standard problem, and we use CG FEM to compute the solution. We obtain an adapted solution of the cutting problem – only the specific cut scenario mentioned above can be simulated without computational expensive changes to the mesh.

The first check of our implementation of X-FEM is visually accurate, as can be seen in figure 8.



(a) the complete mesh of the test object

(b) partially cut and completely cut elements above the cut, coloured in orange and green

Figure 8: Visualization of the test case with an mesh with $N_{\mathcal{T}} = 1810$ and $N = 595$.

In the next chapter, we confirm those results by applying the error measure introduced above.

5.2 Convergence analysis

We evaluate the results of the implementation of the X-FEM in C++ by means of the reference solution of CG FEM. We compute the RMS error for different numbers of degrees of freedom (DoFs) and compare it with the error of the results obtained using CG FEM, see figure 9: on the x_0-axis the number of DoFs and on the x_1-axis the RMS error - in red the error of CG FEM, in blue the error of our implementation of X-FEM based on C++. The convergence analysis can be reproduced using the open source version of our implementation.

In a mesh of a domain Ω that consists of equidistant points, there is a close correlation between

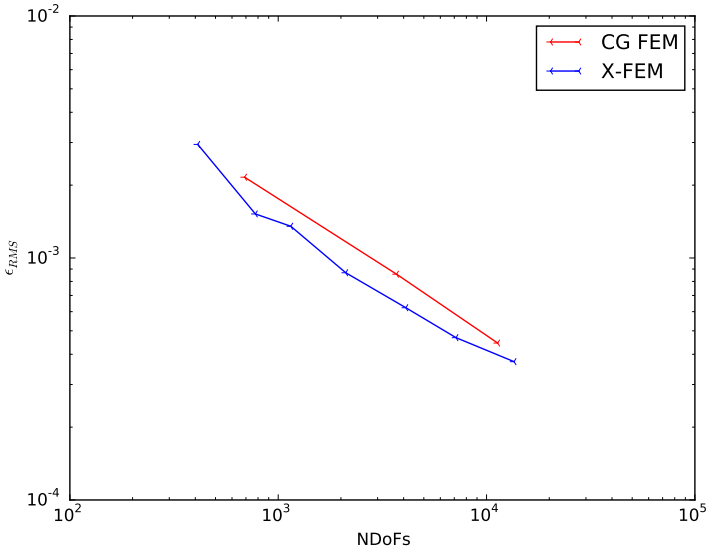


Figure 9: Development of the RMS errors using X-FEM and CG FEM

the number of DoFs and the greatest length h of an edge of a tetrahedron that belongs to the mesh. Therefore most of the evaluations of conventional finite element methods contain the development of the error dependent on h instead of the number of DoFs. In the X-FEM, the number of DoFs depends on the cut as well, so it is difficult to establish a relation between the number of DoFs and h. Furthermore, the computation time depends on the number of DoFs, i.e. knowing about the number of DoFs helps to better decide whether a result can be obtained in real time. Therefore we restricted ourselves to only displaying the error dependent on the number of DoFs.

In the considered test case the error of our method decreases monotonously with the increase of the number of DoFs. The error of the solution obtained with our implementation stays less than the error of an adapted FEM mesh. However, the convergence of the X-FEM implementation seems to be weaker than the convergence of the standard FEM.

The phenomenom of the weak convergence rate, has been discovered and discussed in several papers. Chessa et al [3] relate this to the blending elements, which are in between enriched and non-enriched elements, i.e., elements that have enriched and non-enriched nodes. They claim and show in this paper that the violation of the partition of unity in the blending elements negatively impacts the approximations' accuracy. The deterioration of the accuracy in the blending elements weakens the improvement of accuracy that comes from the enrichment. A constant space of enrichment minimizes this negative effect of the blending elements. The general idea of this technique and of other similar ones, namely the prevention of the deterioration in convergence rate, can be found in [16].

Some of the meshes that we used for the evaluations have nodes that are close to the cut of the test problem. Cuts like those can yield *sign enriched* nodes with shape functions having a relatively small support above or below the cut compared to the volume of the elements that are attached to the node, i.e.,

$$\exists k \in \{N_{\mathcal{P}}, \ldots, N_{\mathcal{P}} + N_S - 1\} : \left(\frac{\lambda_L \left(\mathrm{supp}(\Phi_k) \cap \Omega_+\right)}{\lambda_L \left(\bigcup_{T \in \mathcal{T}: \boldsymbol{P}_k \in T} T\right)} \approx 0 \right) \vee \left(\frac{\lambda_L \left(\mathrm{supp}(\Phi_k) \cap \Omega_-\right)}{\lambda_L \left(\bigcup_{T \in \mathcal{T}: \boldsymbol{P}_k \in T} T\right)} \approx 0 \right)$$

Daux et al [6] showed that such nodes lead to singularities, if they are not treated in the right way. This has not been a problem for our calculations, but it could trigger weak convergence rate.

At this point, we underline that in our simulation the grid of the beam is arbitrary and not adjusted to the cut, i.e., the results we obtain by applying the X-FEM are average results. Still, the accuracy of our solution surpasses the solutions of the CG FEM, which have been specifically adapted to the problem.

5.3 Cut through complex geometry

In order to demonstrate the flexibility of the approach, we perform a sinusoidal cut through a very low resolution liver geometry (Fig. 10, the example is provided in the git repository). The undeformed finite element model consists of 888 linear tetrahedra and 397 nodes, i.e. 1191 degrees of freedom. An additional 303 degrees of freedom are added in order to model the cut.

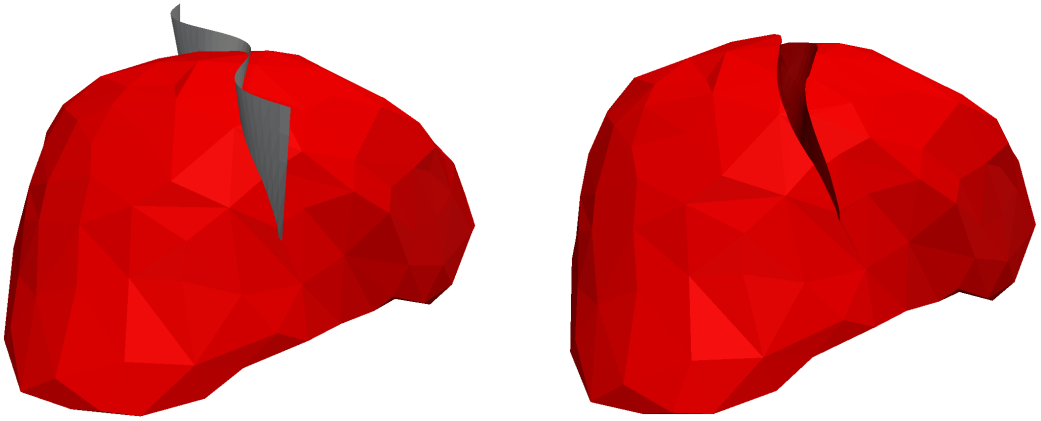


Figure 10: A liver with 888 linear tetrahedral elements is cut by a sinusoidal blade.

5.4 Conclusion

The evaluation showed, that the X-FEM provides an accurate solution of the cutting problem. The solution of the method surpasses the solution of the conventional finite element method that has been adapted to the specific problem.

In our implementation an arbitrary mesh for the object can be chosen without restricting the cut. Opposed to this, in the CG FEM the mesh has to be adapted when we apply a cut, which already is a complex task for simple cut geometries. An adaption of the mesh is computationally expensive and therefore has a negative impact on the real-time capacity of the simulation. Despite of the computational cost for the adjustment of the grid, the accuracy of the solution of the CG FEM is worse than the accuracy of our implementation.

We discretize the cut using triangles. With triangles we can represent any arbitrary 2D object, which means arbitrary cuts can be discretized. Therefore, the X-FEM can handle cuts that either cannot be calculated using the CG FEM at all, or only at high computational cost.

6 Summary and outlook

In the context of this work, the cutting problem which arises when deforming and cutting soft tissue under the influence of force and displacement boundary conditions, has been modelled, discretized, solved and simulated applying the eXtended finite element method (X-FEM).

We presented the cutting problem in elasticity by means of its differential and its variational formulation. We discretized the object and the cut to allow for finding a solution using numerical methods. We adjusted the mesh to the topological changes that arise from the intersections of the meshes of the object and the cut and defined branch and sign enriched nodes. Then the shape functions for the conventional and the X-FEM were defined. Considering approximative discrete function for the variational formulation, we obtained an integral equation. An adapted numerical integration finally yields the discretized displacement of the cutting problem as the solution of a system of linear equations.
In the case of an implementation of the dynamic problem, we presented the geometric precomputations. Furthermore, our method was implemented in C++ using methods provided by the open source framework CGAL. The implementation allows a three dimensional simulation of an arbitrary cutting problem without adaptation of the grid of the object to the grid of the cut.
For the evaluation of our C++ implementation, we compare our simulation of a stretched beam to a reference solution, that was obtained with CG FEM. The analysis proved the convergence of the implemented method. In this simple example, our method has a higher accuracy than the solutions of the conventional finite element method, which are obtained by perfectly aligning the grid of the object to the cut. In problems with higher complexity, conventional methods either fail or get computationally expensive, whereas the computation of solutions based on our method is numerically stable.
In the future, we want to simulate models with higher degrees of freedom without a drawback in computation time. This could be achieved by replacing the oversampling integration method by another integration technique. We suggest to find a polynomial that replaces or approximates the asymptotic crack tip functions and to apply the idea of Mirtich [14] – he proposed to replace an integral over a volume by an integral over a face and then by an integral over a line. This approach allows very fast computations on complex volumes like tetrahedra separated by a triangulated cut. An application of the co-rotational finite element method combined with the X-FEM would eliminate the unnatural deformation that arises because of ghost forces, see [15]. Furthermore, a dynamic declaration of the cut in each time step would allow for progressive cutting, which represents another challgenging task in the context in surgery simulation applications. Finally, allowing for non-zero force boundary conditions on the cut surface would enable the simulation of collisions of the body parts above and below the cut.

7 Acknowledgment

This was carried out with the support of the German Research Foundation (DFG) within the projects A01 and I03 of the SFB/TRR 125 'Cognition-Guided Surgery'.

References

[1] K.J. Bathe. *Finite Element Procedures.* Prentice Hall, 1996.

[2] Daniel Bielser, Pascal Glardon, Matthias Teschner, and Markus Gross. A state machine for real-time cutting of tetrahedral meshes. *Graphical Models*, 66(6):398–417, 2004.

[3] J. Chessa, H. Wang, and T. Belytschko. On the construction of blending elements for local partition of unity enriched finite elements. *International journal for numerical methods in engineering*, 57:1015–1038, 2003.

[4] 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.

[5] H. Courtecuisse, H. Jung, J. Allard, C. Duriez, D.Y. Lee, and S. Cotin. Gpu-based real-time soft tissue deformation with cutting and haptic feedback. *Progress in biophysics and molecular biology*, 103(2):159–168, 2010.

[6] C. Daux, N. Moes, J. Dolbow, N. Sukumar, and T. Belytschko. Arbitrary branched and intersecting cracks with the extended finite element method. *International journal for numerical methods in engineering*, 48:1741–1760, 2000.

[7] C. Dick, J. Georgii, and R. Westermann. A Hexahedral Multigrid Approach for Simulating Cuts in Deformable Objects. *Visualization and Computer Graphics, IEEE Transactions on*, (99):1–1, 2010.

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

[9] Jan Hegemann, Chenfanfu Jiang, Craig Schroeder, and Joseph M Teran. A level set method for ductile fracture. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 193–201. ACM, 2013.

[10] L. Jerabkova, G. Bousquet, S. Barbier, F. Faure, and J. Allard. Volumetric modeling and interactive cutting of deformable bodies. *Progress in biophysics and molecular biology*, 2010.

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

[12] P. Kaufmann, S. Martin, M. Botsch, E. Grinspun, and M. Gross. Enrichment textures for detailed cutting of shells. *ACM Transactions on Graphics (TOG)*, 28(3):50, 2009.

[13] A. Mazura and S. Seifert. Virtual cutting in medical data. *Studies in health technology and informatics*, 39:420, 1997.

[14] Brian Mirtich. Fast and accurate computation of polyhedral mass properties. 1996.

[15] Nicolai Schoch, Stefan Suwelack, Rdiger Dillmann, and Vincent Heuveline. Simulation of surgical cutting of soft tissue using the x-fem. *Preprint Series of the Engineering Mathematics and Computing Lab*, 0(04), 2013.

[16] Y. Shen and A. Lew. An optimally convergent discontinuous galerkin-based extended finite element method for fracture mechanics. *International journal for numerical methods in engineering*, 82:716–755, 2010.

[17] Eftychios Sifakis, Kevin G Der, and Ronald Fedkiw. Arbitrary cutting of deformable tetrahedralized objects. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 73–80. Eurographics Association, 2007.

[18] Lara M. Vigneron. *FEM/XFEM-Based Modeling of Brain Shift, Resection, and Retraction for Image-Guided Surgery*. PhD thesis, University of Lige,, 2009.

[19] L.M. Vigneron, S.K. Warfield, P.A. Robe, and J.G. Verly. 3D XFEM-based modeling of retraction for preoperative image update. *Computer Aided Surgery*, 16(3):121–134, 2011.

[20] Jun Wu, Christian Dick, and Rüdiger Westermann. Efficient collision detection for composite finite element simulation of cuts in deformable bodies. *The Visual Computer*, 29(6-8):739–749, 2013.

Preprint Series of the Engineering Mathematics and Computing Lab

recent issues

No. 2014-01 Martin Wlotzka, Vincent Heuveline: A parallel solution scheme for multiphysics evolution problems using OpenPALM

No. 2013-04 Nicolai Schoch, Stefan Suwelack, Ruediger Dillmann, Vincent Heuveline: Simulation of Surgical Cutting in Soft Tissue using the Extended FEM

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 N_2O 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

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

www.emcl.iwr.uni-heidelberg.de