JÖNKÖPING UNIVERSITY
School of Engineering

Doctoral Thesis

Finite element methods for surface problems

Mirza Cenanovic

Jönköping University
School of Engineering
Dissertation Series No. 022 • 2017

Doctoral Thesis in Machine Design

Finite element methods for surface problems
Dissertation Series No. 022

Prepared with LyX and LaTeX

Published by
Department of Product Development
School of Engineering, Jönköping University
P.O. Box 1026
SE-551 11 Jönköping, Sweden
Tel.: +46 36 10 10 00
www.ju.se

Printed by Ineko AB, year 2017

ISBN 978-91-87289-23-1

What I cannot create, I do not understand

- Richard Feynman

Abstract

The purpose of this thesis is to further develop numerical methods for solving surface problems by utilizing *tangential calculus* and the *trace finite element method*. Direct computation on the surface is possible by the use of tangential calculus, in contrast to the classical approach of mapping 2D parametric surfaces to 3D surfaces by means of differential geometry operators. Using tangential calculus, the problem formulation is only dependent on the position and normal vectors of the 3D surface. Tangential calculus thus enables a clean, simple and inexpensive formulation and implementation of finite element methods for surface problems. Meshing techniques are greatly simplified from the end-user perspective by utilizing an unfitted finite element method called the Trace Finite Element Method, in which the basic idea is to embed the surface in a higher dimensional mesh and use the shape functions of this background mesh for the discretization of the partial differential equation. This method makes it possible to model surfaces implicitly and solve surface problems without the need for expensive meshing/re-meshing techniques especially for moving surfaces or surfaces embedded in 3D solids, so called embedded interface problems. Using these two approaches, numerical methods for solving three surface problems are proposed: 1) minimal surface problems, in which the form that minimizes the mean curvature was computed by iterative update of a level-set function discretized using TraceFEM and driven by advection, for which the velocity field was given by the mean curvature flow, 2) elastic membrane problems discretized using linear and higher order TraceFEM, which makes it straightforward to embed complex geometries of membrane models into an elastic bulk for reinforcement and 3) stabilized, accurate vertex normal and mean curvature estimation with local refinement on triangulated surfaces. In this thesis the basics of the two main approaches are presented, some aspects such as stabilization and surface reconstruction are further developed, evaluated and numerically analyzed, details on implementations are provided and the current state of work is presented.

Keywords: trace finite element method, membrane, mean curvature, level-set method

Papers

The following papers constitute the basis of this thesis.

Paper I	Mirza Cenanovic, Peter Hansbo, Mats G. Larson Minimal surface computation using finite element method on an embedded surface
Paper II	Mirza Cenanovic, Peter Hansbo, Mats G. Larson Cut finite element modeling of linear membranes
Paper III	Mirza Cenanovic, Peter Hansbo, Mats G. Larson Finite element procedures for computing normals and mean curvature on triangulated surfaces and their use for mesh refinement
Paper IV	Mirza Cenanovic Numerical error estimation for a TraceFEM membrane and distance function on P1 and P2 tetrahedra

Contribution to co-authored papers

All but one paper are co-authored with Hansbo and Larson, who are responsible for the theoretical framework.

The contribution by the author of this thesis is listed below.

Paper I	Took part in planning the paper. Made the numerical implementations. Carried out the numerical simulations. Took part in writing the paper.
Paper II	Took part in planning the paper. Made the numerical implementations. Carried out the numerical simulations. Took part in writing the paper.
Paper III	Took equal part in planning the paper. Made the numerical implementations. Carried out the numerical simulations. Wrote major parts of the paper.

Acknowledgements

The work presented in this thesis was carried out during the years 2012-2017 as part of a research project at the department of Product development at School of Engineering, Jönköping University. The work is funded by the Swedish Research Council (Grant No. 2011-4992).

First I would like to thank my supervisor Professor Peter Hansbo for all the support, discussions and patience. I also want to thank my co-supervisor Associate Professor Kent Salomonsson for his support, discussions and guidance. The expert knowledge from each of their fields and pedagogical skills provided a superb and relaxed learning environment for which I am deeply grateful for. I would like to thank all my colleagues at the department for the easygoing atmosphere and fruitful discussions during coffee breaks. Lastly, I would like to thank my family for all the support, especially during the months of the final stages of completing this thesis.

Mirza Cenanovic
March 2017

Contents

Introduction

Introduction and motivation

Surface problems are found in many places, such as manufacturing industry, architecture and computer graphics. Problem areas include, for instance, sheet metal forming which can be modeled using shell theory, membranes such as sails and thin structures, see e.g., [23], heat transfer across a curved geometry, computing the distance between two points on a curved surface (also called geodesic distance field, see, e.g., [14]). Other problem areas include curvature driven problems and composite materials where thin layers add stiffness to the bulk (sandwich constructions in airplane industry and glue laminated timber).

Surface problems. Traditionally, three dimensional surface problems were modeled in a two dimensional parameter space and mapped to the real surface in three dimensions [13, 12], also known as the parameterization of a surface or parametric approach to surface problems. This requires the use of an exact surface representation and additional differential operators that are defined on a curved space using base vectors and Christoffel symbols. This might be computationally tedious, cumbersome to implement and expensive to compute. Besides, an exact surface representation is not always available; a CAD surface is typically defined by a set of parameterized surface patches that are not necessary continuous across the interfaces between the two surfaces. Another classic engineering approach is to model the surface as a set of flat triangles and rotate each into three dimensional space [50]; these formulations are however expressed in a discrete setting as matrix operations.

An approach to modeling surface partial differential equations (PDEs) without the use of parametric mapping was introduced for surface stresses by Gurtin and Murdoch [27]. This so called *tangential approach* (also known as *tangential differential calculus*) was first used in finite element methods by Dziuk [21] for the discretization of the Laplace-Beltrami operator on a meshed surface, where (using a signed distance function) a tangential gradient was introduced and the resulting

numerical scheme was rather clean and simple. The same geometric differential calculus is used by Delfour and Zolésio in [16, 17, 19], where, again, a signed distance function is used to represent the surface and from which the tangential differential operator is derived and used to create a linear shell model without parametric mapping. This approach was followed by Hansbo and Larson in [29], where a FEM was created for a general curved linear membrane model, and by Hansbo, Larson and Larsson in [28] for large deformations theory. A recent overview of FEM for surface PDEs is given by Dziuk and Elliot in [22], where the tangential approach is applied on a large set of surface PDEs and discussed in the paper and references therein.

There are a couple of fundamental properties of surfaces that are extremely important in computer graphics and computer vision: surface normals and surface curvature. With the increase in computing power we see an increase in need for accurate approximations of these fundamental surface properties. Areas such as the gaming industry, film industry and medical image scanning such as CT, MR, 3D ultrasound, all require accurate approximations of surface normals and surface curvature. In case of accurate surface interpolation using methods involving the surface normals, the accuracy of the surface interpolation depends on the accuracy of the surface normal. Traditionally, these have been approximated locally for fast execution times with little regard for accuracy, see e.g., [1, 3, 40, 20, 49, 39, 38, 37, 34, 26].

Unfitted methods. In order to shorten development cycles in industry the CAD, CAE, and structural optimization are required to seamlessly fit together and ultimately iterate with as little human intervention as possible. Thus, to move towards this ultimate goal of automation, we need to have sufficiently accurate and performance efficient tools for representing implicit geometries and employ them in FE methods. Recently, focus has increased on the development of methods which do not require conforming meshes and are defined by the fact that the domain, on which a PDE is to be solved, is completely embedded in a fictitious domain (background mesh) and thus go under the umbrella term *fictitious domain methods* (see e.g., [22, 43] and the references therein). For evolving surface problems a novel FEM was proposed by Olshanskii, Reusken and Grande [44] for elliptic PDEs, where the surface was embedded into a higher dimensional mesh (also called background mesh) and allowed to arbitrarily cut through the mesh. The PDE was then discretized using the shape functions of the background mesh but integrated over the approximated surface. That triggered an avalanche of studies following this new approach and the reader is referred to [43] and Chapter 2 of this thesis for a detailed overview of some recent work on surface and fictitious domain problems, called the *Trace Finite Element Method* (TraceFEM). This approach provides robust and general methods for dealing with a surface geometry that does not necessary respect the background mesh (the surface is allowed to cut through the background mesh), see e.g. [9, 30, 5, 43]. A lot of current work is being put into numerical integration of implicitly defined domains, see e.g., [43, 45, 42, 35, 24, 25]. Since the introduction of unfitted finite element methods a lot of work has been done to improve the numerical stability and conditioning of the linear system, see e.g.,

[4, 7, 8, 9, 6], in particular see the recent overview by Olshanskii and Reusken on alternative methods in [43].

The work done in this thesis further develops methods for solving surface PDEs using the tangential calculus approach within the TraceFEM framework, see Papers 1, 2 and 4. In Paper 3 we use the recently developed stabilization method on triangular meshes.

Aim and limitations

The aim of this work is to apply the tangential calculus in combination with TraceFEM to a set of surface problems in order to improve the computer aided engineering modeling of surface problems and in particular moving surfaces and embedded surfaces. The goal of this project is to increase the knowledge within the field of surface PDEs using the tangential approach as well as TraceFEM with regards to computational techniques. This is achieved by further developing aspects of the TraceFEM in the context of the previously mentioned surface problems, providing numerical results and analysis of the underlying computational methods with some comparisons to previously proposed methods. The purpose of this thesis is to give a brief overview of the ideas of the various approaches used in the papers and to show the details of the implementations. Since the implementations are done in 3D, the implementations and their details are limited to small proof of concept scripts which are implemented in the high level language MATLAB[1] and need rework (with respect to performance) for production use. The algorithms provided in the Papers are kept general and without extensive implementation details. With this in mind no complexity analysis was done since the aim was convergence analysis, readability, share-ability and short implementation times. It is the hope of the author that this thesis will be useful to anyone who wants to implement the ideas of the approaches discussed within this thesis. Some routines used in this thesis have been made available on bit-bucket[2], albeit with little documentation.

[1] http://mathworks.com/products/matlab/
[2] https://bitbucket.org/jthsimopt/

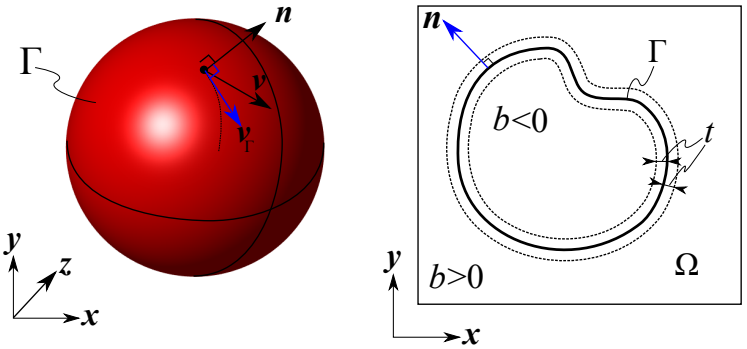
1 Tangential Calculus

CHAPTER INTRODUCTION

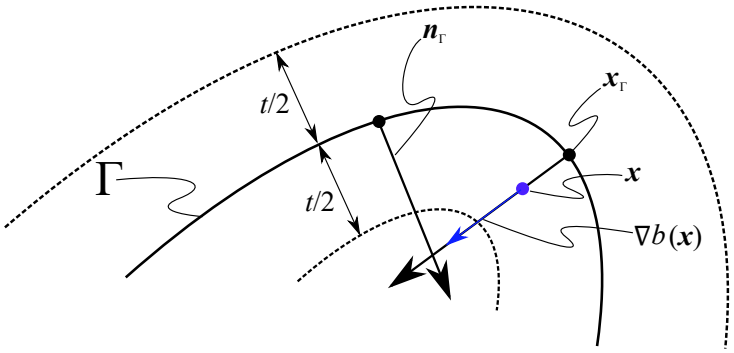
This chapter introduces the concept of *tangential calculus* which is used for the modeling of surface problems directly in Cartesian coordinates. An overview of this approach is given. The two main surface representations are introduced.

1.1 The tangential operator

This section gives an overview of what is here called the tangential approach to surface problem modeling, which was introduced in computational practice by Dziuk in [21] and analyzed for shells by Delfour and Zolésio in [16, 17, 19] and adapted by Hansbo and Larson in [29] for a general curved membrane model. Traditionally, surface problems were modeled in a parametric setting and mapped to Cartesian three dimensional space using various differential operators, see [13, 12]. The tangential approach makes it possible to work in Cartesian space directly by use of the signed distance function of an implicit surface. This is due to the property that the tangential derivative of a function, at the surface, is the tangential projection of its three dimensional Cartesian gradient [17].



(a) 2D surface in 3D (b) Surface manifold



(c) Tubular neighborhood Ω_t

Figure 1.1: Surface Manifold

Consider a smooth d-dimensional surface Γ, see Figure 1.1, embedded in $\mathbb{R}^{d+1}$, where d is typically 1 or 2, and contained by Ω which is a subset of $\mathbb{R}^{d+1}$. The shortest distance from a point $\boldsymbol{x} \in \Omega$ to Γ is given by a signed distance function $b : \mathbb{R}^{d+1} \to \mathbb{R}$ which is defined by,

$$\begin{aligned} b(\boldsymbol{x}) &< 0 \quad \text{in } \Omega_-, \\ b(\boldsymbol{x}) &= 0 \quad \text{on } \Gamma, \\ b(\boldsymbol{x}) &> 0 \quad \text{in } \Omega_+, \end{aligned}$$

and $\Omega_+ \in \Omega$ is the manifold "outside" of Ω, i.e., in the direction of the normal to Γ, $\Omega_- \in \Omega$ is the manifold "inside" of Γ and $\Omega = \Omega_- \cup \Omega_+ \cup \Gamma$. Note that b is not necessarily differentiable everywhere in Ω, and for some points $\boldsymbol{x} \in \Omega$ there might exist more than one surface point $\boldsymbol{x}_\Gamma$ that has the same distance to $\boldsymbol{x}$.

Remark 1. Take the torus as an example, see Figure 1.2. The distance function for the torus is given by

$$b_{\text{torus}}(x, y, z) = \left(R - \sqrt{x^2 + y^2}\right)^2 + z^2 - r^2 \tag{1.1}$$

On the center line of the tube the gradient of the distance is a zero vector since there exist an

infinite amount of closest points to the surface. This can be shown by computing

$$\nabla b_{\text{torus}}(x,y,z) = \left[-\frac{2x\left(R-\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}}, \quad -\frac{2y\left(R-\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}}, \quad 2z \right] \tag{1.2}$$

and evaluating at, e.g., $\nabla b_{\text{torus}}(1,0,0) = [0,0,0]$, similarly for, e.g., $\nabla b_{\text{torus}}(0,0,0)$ we can see that the gradient is not defined since we get division by zero. □

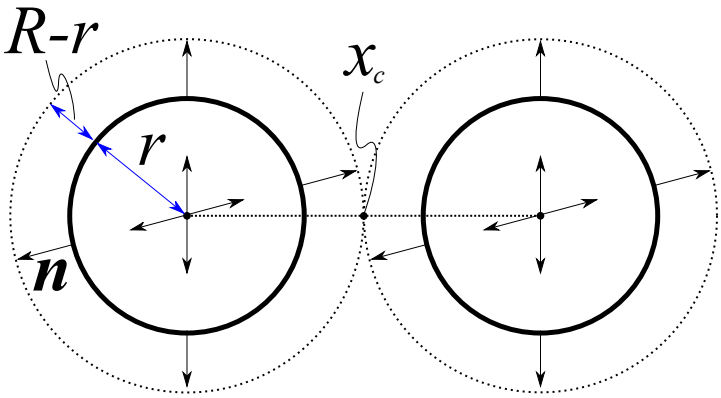


Figure 1.2: Cross section view of a torus with tube radius r and major radius R. The gradient of a distance function for this torus is not defined in the tube center. The tubular neighborhood is bounded by the dotted circle, i.e., outside of this area $\nabla b(\boldsymbol{x})$ is not guaranteed to be unique and may be undefined.

It is thus important, for a small $t > 0$ around Γ, to define a neighborhood Ω_t (defined as the space between the dotted lines in Figure 1.1c) such that $b(\boldsymbol{x})$ is differentiable everywhere in Ω_t. Choose t small enough such that every point $\boldsymbol{x} \in \Omega_t$ is intersected by one unique surface normal and such that no surface normals are allowed to intersect each other inside of Ω_t, see Figure 1.1c. The neighborhood around Γ can be defined as

$$\Omega_t = \{\boldsymbol{x} \in \mathbb{R}^{d+1} : |b(\boldsymbol{x})| \leq t\}, \tag{1.3}$$

and with $|\nabla b| = 1$ we have

$$\nabla b(\boldsymbol{x}_\Gamma) = \boldsymbol{n}(\boldsymbol{x}_\Gamma). \tag{1.4}$$

Inside Ω_t the normal vector to Γ coincides with $\nabla b(\boldsymbol{x})$ and thus the nearest point projection map $\boldsymbol{p} : \Omega_t \to \Gamma$ is given by

$$\boldsymbol{p}(\boldsymbol{x}) = \boldsymbol{x} - b(\boldsymbol{x})\nabla b(\boldsymbol{x}) \tag{1.5}$$

The Jacobian matrix is given by differentiating $\boldsymbol{p}(\boldsymbol{x})$, the first component yields

$$\frac{\partial}{\partial x_1} p_1 = 1 - \frac{\partial}{\partial x_1}\left(b(\boldsymbol{x})\frac{\partial b(\boldsymbol{x})}{\partial x_1}\right) = 1 - \frac{\partial b(\boldsymbol{x})}{\partial x_1}\frac{\partial b(\boldsymbol{x})}{\partial x_1} - b(\boldsymbol{x})\frac{\partial^2 b(\boldsymbol{x})}{\partial x_1^2}, \tag{1.6}$$

or

$$\boldsymbol{I} - \nabla b(\boldsymbol{x}) \otimes \nabla b(\boldsymbol{x}) - b(\boldsymbol{x}) D^2 b(\boldsymbol{x}) \tag{1.7}$$

where $b(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in \Gamma$ so $b(\boldsymbol{x})D^2 b(\boldsymbol{x}) = 0$ on the surface. Thus for $\boldsymbol{x} \in \Gamma$, the *linear projector onto the tangent plane* at $\boldsymbol{p}(\boldsymbol{x})$ is given by

$$D\boldsymbol{p}(\boldsymbol{x}) = \boldsymbol{I} - \nabla b(\boldsymbol{x}) \otimes \nabla b(\boldsymbol{x}) =: \boldsymbol{P}_\Gamma(\boldsymbol{x}), \tag{1.8}$$

where $\otimes$ denotes the tensor product ($(\boldsymbol{a} \otimes \boldsymbol{b})_{ij} = a_i b_j$).

For a given function $u : \Gamma \to \mathbb{R}$ we can assume that there exists an extension u^e of u in Ω_t such that $u^e = u \circ \boldsymbol{p}$, where the symbol $\circ$ denotes the function composition $(a \circ b)(x) = a(b(x))$. The tangential gradient of u on Γ is thus defined by

$$\nabla_\Gamma u := \nabla u^e|_\Gamma - \nabla u^e \cdot \nabla b \; \nabla b = (\boldsymbol{P}_\Gamma \nabla u^e)|_\Gamma \quad \text{on } \Gamma, \tag{1.9}$$

and

$$\boldsymbol{n} \cdot \nabla_\Gamma u = \nabla b \cdot \nabla_\Gamma u = 0 \tag{1.10}$$

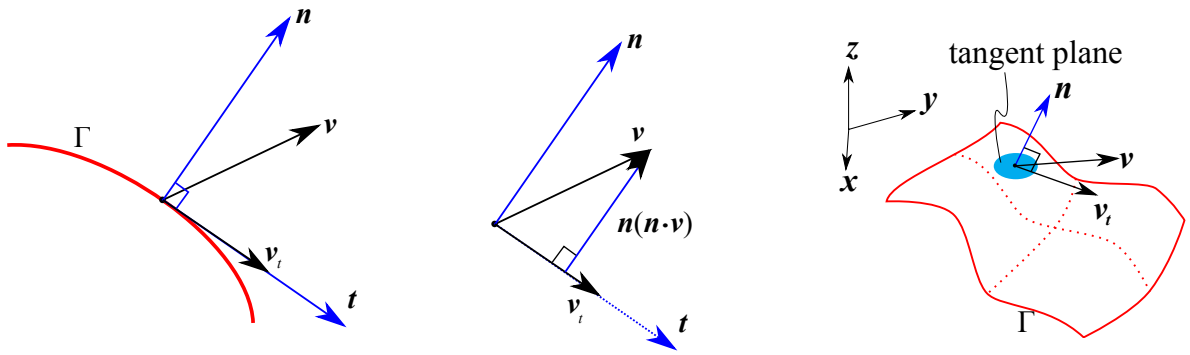
where the subscript Γ of ∇_Γ implies that the partial derivative components are with respect to the surface points, $\boldsymbol{x}_\Gamma$. The tangential (or surface) gradient in (1.9) is defined using the full Cartesian gradient and removing the "out of plane" contribution. It can be shown that the gradient of the extension of u coincides with the tangential gradient of u on Γ

$$\nabla(u \circ \boldsymbol{p}) = (\boldsymbol{I} - bD^2 b)\nabla_\Gamma u \circ \boldsymbol{p} \text{ and } \nabla(u \circ \boldsymbol{p})|_\Gamma = \nabla_\Gamma u, \tag{1.11}$$

see [18, Chapter 9, Section 5] for details and proofs. Thus the resulting tangential gradient is independent of the choice of the extension u^e and in what follows no distinction is made between u and its extension u^e.

The tangential operator $\boldsymbol{P}_\Gamma$ is the main differential geometric tool used in the tangential approach and is used extensively in this thesis on all surface problems. For the convenience of notation, the subscript of $\boldsymbol{P}_\Gamma$ will frequently be omitted. Note that the operator is natural in tensor analysis, see

e.g. [32], where it is defined as the projection of a vector valued function $\boldsymbol{v}$ onto the plane defined by the normal $\boldsymbol{n}$ see Figure 1.3.



(a) 2D case (b) Defining $\boldsymbol{v}_n = \boldsymbol{n}(\boldsymbol{n}\cdot\boldsymbol{v})$ (c) 3D case

Figure 1.3: Projection of $\boldsymbol{v}$ onto the surface Γ

Using the definition in Figure 1.3b, we have

$$\boldsymbol{v}_t = \boldsymbol{v} - \boldsymbol{v}_n = \boldsymbol{v} - \boldsymbol{n}(\boldsymbol{n}\cdot\boldsymbol{v}). \tag{1.12}$$

Recalling the tensor product ($(\boldsymbol{a}\otimes\boldsymbol{b})_{ij} = a_i b_j$), we can use the tensor product rule

$$(\boldsymbol{a}\otimes\boldsymbol{b})\boldsymbol{c} = \boldsymbol{a}(\boldsymbol{b}\cdot\boldsymbol{c}), \tag{1.13}$$

(which is what Dziuk used in [21]) and get

$$\boldsymbol{v}_t = \boldsymbol{v} - (\boldsymbol{n}\otimes\boldsymbol{n})\boldsymbol{v}, \tag{1.14}$$

where $\boldsymbol{v}$ is linearly transformed along the direction $\boldsymbol{n}$ by the second order tensor $(\boldsymbol{n}\otimes\boldsymbol{n}) =: \boldsymbol{P}_{||}$. This can be further rewritten into

$$\boldsymbol{v}_t = (\boldsymbol{I} - \boldsymbol{n}\otimes\boldsymbol{n})\boldsymbol{v} = \boldsymbol{P}\boldsymbol{v}, \tag{1.15}$$

where $\boldsymbol{P}$ is a second order projection tensor that maps onto the tangent plane of Γ and $\boldsymbol{P}_{||}$ maps onto the direction of $\boldsymbol{n}$.

Remark 2. A detailed look at the tangential part of a vector $\boldsymbol{v}$ at a point on a surface where $\boldsymbol{n}$=(1,0,0). Let $\boldsymbol{v} = (v_x, v_y, v_z)$ then,

$$\boldsymbol{v}_t = \boldsymbol{P}\boldsymbol{v} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} 0 \\ v_y \\ v_z \end{pmatrix}. \tag{1.16}$$

The x component of $\boldsymbol{v}$ has been eliminated by the projection operator and thus the other components are all in-plane, i.e. $\boldsymbol{n} \cdot \boldsymbol{v}_t = 0$ holds.

Another detailed look at the tangential part of a tensor $\boldsymbol{V}$ at a point on a surface where $\boldsymbol{n} = (0, 1, 0)$. Let

$$\boldsymbol{V} = \begin{pmatrix} V_{xx} & V_{xy} & V_{xz} \\ V_{yx} & V_{yy} & V_{yz} \\ V_{zx} & V_{zy} & V_{zz} \end{pmatrix}, \qquad \boldsymbol{P} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{1.17}$$

then

$$\boldsymbol{V}_t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_{xx} & V_{xy} & V_{xz} \\ V_{yx} & V_{yy} & V_{yz} \\ V_{zx} & V_{zy} & V_{zz} \end{pmatrix} = \begin{pmatrix} V_{xx} & V_{xy} & V_{xz} \\ 0 & 0 & 0 \\ V_{zx} & V_{zy} & V_{zz} \end{pmatrix}. \tag{1.18}$$

Note that the row corresponding to the normal direction is eliminated. In some cases it is required that both the rows and columns are eliminated so that no components of the tensor are out of plane. This can be accomplished by applying the operator to both sides of the tensor, i.e.

$$\boldsymbol{V}_t^P = \boldsymbol{P}\boldsymbol{V}\boldsymbol{P} = \begin{pmatrix} V_{xx} & V_{xy} & V_{xz} \\ 0 & 0 & 0 \\ V_{zx} & V_{zy} & V_{zz} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} V_{xx} & 0 & V_{xz} \\ 0 & 0 & 0 \\ V_{zx} & 0 & V_{zz} \end{pmatrix}, \tag{1.19}$$

where the superscript P denotes that the operator is applied on both sides. Note that no components exist in $\boldsymbol{V}_t^P$ that correspond to the normal direction. □

1.1.1 The discrete tangential projection

In the discrete setting where the surface is discretized by a piecewise polynomial function, the projection operator will depend on the discretization such that $\boldsymbol{P} - \boldsymbol{P}_h \neq \boldsymbol{0}$, where $\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{n} \otimes \boldsymbol{n}$ and $\boldsymbol{P}_h = \boldsymbol{I} - \boldsymbol{n}_h \otimes \boldsymbol{n}_h$, see Figure 1.4. This means that we have have introduced another discretization error through the projection operator. For an in depth discussion on this error see [9].

1.2 Surface representation

Surfaces can be represented in various ways; in the setting of solving partial differential equations on surfaces using the finite element method we have two types of surfaces to consider, explicit- and implicit surfaces. The former is a surface discretized using polygons in 3D or line segments in 2D, usually from a parametric (CAD) representation, see Figure 1.5. If the explicit representation is of good quality, i.e., if the elements have a good aspect ratio and are oriented in the same direction, then the mesh can be used in the FEM simulation. In many problems re-meshing is needed as the surface undergoes large deformation such that the mesh quality suffers. Re-meshing techniques

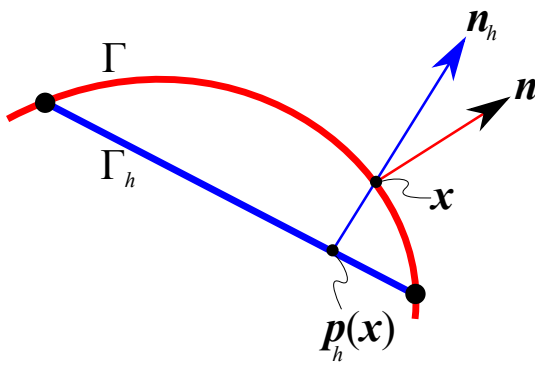


Figure 1.4: Discrete normal $\boldsymbol{n}_h$ compared to exact surface normal $\boldsymbol{n}$

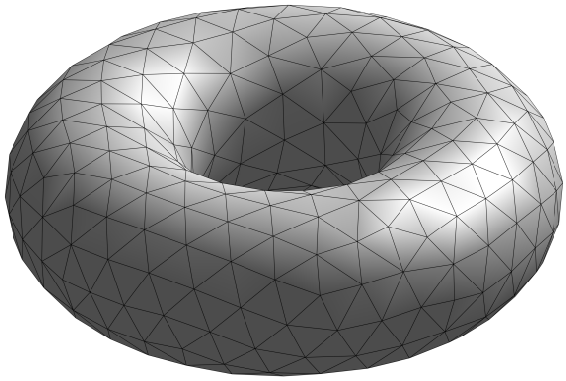


Figure 1.5: Explicit surface example, a triangulated surface of a torus.

are quite expensive and add complexity. Another drawback of computational methods requiring explicit surfaces is the computational cost of preprocessing of raw data, usually point cloud data.

1.2.1 Implicit surfaces

Implicit surfaces can be used to represent complex evolving surfaces without the need for classical re-meshing techniques. The implicit surface is defined by a level-set function ϕ and the surface Γ is represented by the zero level-set of ϕ, i.e., by solving $\phi(\boldsymbol{x}) = 0$. The surface is evolved through a domain by advection of ϕ (using the material derivative). There are a number of ways to generate an implicit surface for analysis. An implicit surface can be approximated from a CAD surface or from point cloud data using surface reconstruction techniques, see, e.g., [2, 11]. Another way is to use analytical implicit surfaces descriptions, see e.g., [9] for an overview. Complex geometries can be created from simple geometries using analytical functions for simple geometries together with Boolean operations, so called constructive solid geometry (CSG) [33], where several simple level-set

functions are combined into one. Using classical CSG, however, the surface is defined by only one level-set function and thus the consequent surface representation will not be able to represent sharp edges and/or corners. This problem can be overcome by not combining the level sets into one, but use them separately to cut the background domain, see e.g., [41].

Simple boundaries can be created by embedding the surfaces into the domains such that the boundaries are defined by the borders of the domain. For a more general handling of boundary conditions we could use Nitsche's method, see e.g., [10, 9, 36].

An implicit surface Γ is visualized using the discrete faces that are generated by evaluating the level set function ϕ in the nodes of a mesh and finding the root along the faces of each background element, see Figure 1.6, where the level-set function for the cylinder is given by

$$\phi(x, y, z) = \sqrt{(x - x_c)^2 + (y - y_c)^2} - r \tag{1.20}$$

where $[x_c, y_c]$ is the center of the cylinder and for the level-set function for the oblate spheroid is given by

$$\phi(x, y, z) = x^2 + y^2 + (2z)^2 - 1, \tag{1.21}$$

and the level-set function for the torus in given by (1.1).

1.3 Parametric mapping

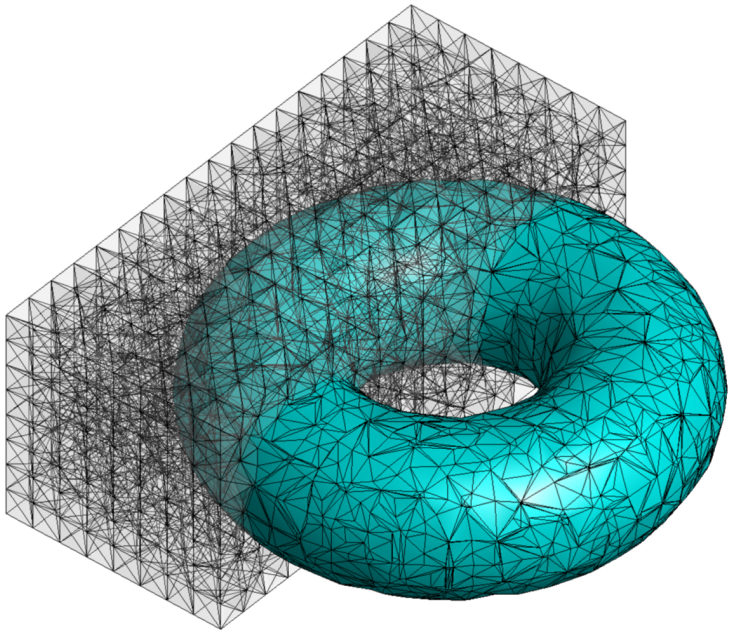
In order to compute surface derivatives on a triangulated surface, i.e., a surface consisting of a set of polygons, we need to define a map. Let $\mathcal{T}_h := \{T_m\}$ define a set of shape regular, conforming polygons of polynomial order m defining the discrete surface Γ_h. In order to define a map $\boldsymbol{M} : (\xi, \eta) \to (x, y, z)$ from a reference polygon in the local coordinate system (ξ, η) to T_m in the physical coordinate system (x, y, z) we define the surface point $\boldsymbol{x}_\Gamma = \boldsymbol{x}_\Gamma(\xi, \eta)$. For a solid object of thickness t represented by the surface, a point inside the object but outside of the surface can then be defined by extending the surface point along the normal

$$\boldsymbol{x}(\xi, \eta, \zeta) = \boldsymbol{x}_\Gamma(\xi, \eta) + \zeta \boldsymbol{n}(\xi, \eta), \tag{1.22}$$

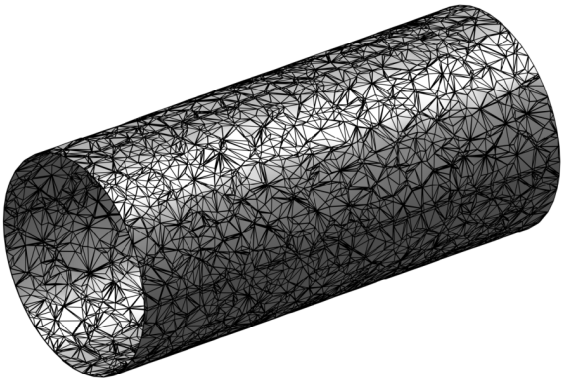
where ζ is chosen such that $-t/2 \leq \zeta \leq t/2$. Next the geometrical finite element parameterization of the surface is given by

$$\boldsymbol{x}_{\Gamma,h}(\xi, \eta) = \sum_i^n \boldsymbol{x}_i \psi_i(\xi, \eta), \tag{1.23}$$

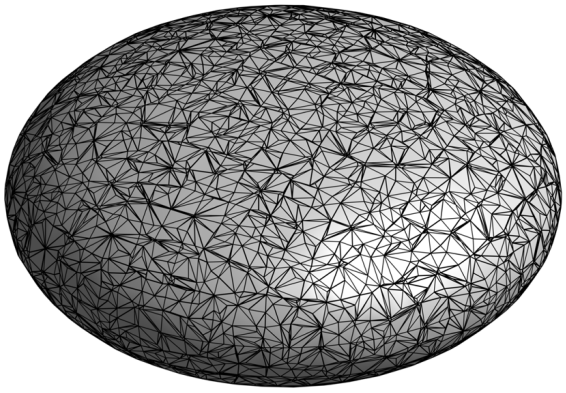
where $\boldsymbol{x}_i$ is one of n nodal points of the polygon T_m of the surface and $\psi_i(\xi, \eta)$ are the finite element shape functions of order m on the reference element. Using this parameterization a point outside of the surface is then approximated by



(a) Embedded Torus



(b) Cylinder



(c) Oblate spheroid

Figure 1.6: Implicit surfaces. The surface Γ is represented by the set of linear triangles Γ_h which is computed by linearly interpolating the nodal valued level-set function ϕ_h along the faces of the background mesh.

$$\boldsymbol{x}(\xi,\eta,\zeta) \approx \boldsymbol{x}_h(\xi,\eta,\zeta) = \boldsymbol{x}_{\Gamma,h}(\xi,\eta) + \zeta \boldsymbol{n}_h(\xi,\eta), \tag{1.24}$$

where

$$\boldsymbol{n}_h = \frac{\frac{\partial \boldsymbol{x}_{\Gamma,h}}{\partial \xi} \times \frac{\partial \boldsymbol{x}_{\Gamma,h}}{\partial \eta}}{\left| \frac{\partial \boldsymbol{x}_{\Gamma,h}}{\partial \xi} \times \frac{\partial \boldsymbol{x}_{\Gamma,h}}{\partial \eta} \right|}. \tag{1.25}$$

For a discussion on the importance of using the discrete normal in a computational context see [29].

The solution field is approximated by

$$\boldsymbol{u} \approx \boldsymbol{u}_h = \sum_i^n \boldsymbol{u}_i \varphi_i(\xi,\eta), \tag{1.26}$$

where φ_i are the shape functions of order q and $\boldsymbol{u}_i$ are the nodal solution values. Note that the shape functions φ_i and ψ_i need not be of the same order, if $q < m$ we have a super-parametric mapping, where the solution field is of order q but the normal field and thus $\boldsymbol{P}$ is of order m, yielding a better approximation. This approach was used in e.g., [29] in order to get a stable solution of the membrane problem.

In order to compute the gradient of the shape function $\nabla\varphi = \left[\frac{\partial \varphi_i}{\partial x}, \frac{\partial \varphi_i}{\partial y}, \frac{\partial \varphi_i}{\partial z}\right]^\intercal$ we start by defining the Jacobian as

$$\boldsymbol{J}(\xi,\eta,\zeta) = \begin{bmatrix} \frac{\partial x_h}{\partial \xi} & \frac{\partial y_h}{\partial \xi} & \frac{\partial z_h}{\partial \xi} \\ \frac{\partial x_h}{\partial \eta} & \frac{\partial y_h}{\partial \eta} & \frac{\partial z_h}{\partial \eta} \\ \frac{\partial x_h}{\partial \zeta} & \frac{\partial y_h}{\partial \zeta} & \frac{\partial z_h}{\partial \zeta} \end{bmatrix}, \tag{1.27}$$

using (1.24) we note that

$$\frac{\partial \boldsymbol{x}_h}{\partial \zeta} = \boldsymbol{n}_h, \tag{1.28}$$

and we have $\frac{\partial \varphi_i}{\partial \zeta} = 0$ and thus

$$\begin{bmatrix} \frac{\partial \varphi_i}{\partial x} \\ \frac{\partial \varphi_i}{\partial y} \\ \frac{\partial \varphi_i}{\partial z} \end{bmatrix} := \boldsymbol{J}^{-1} \begin{bmatrix} \frac{\partial \varphi_i}{\partial \xi} \\ \frac{\partial \varphi_i}{\partial \eta} \\ 0 \end{bmatrix}, \tag{1.29}$$

where the Jacobian on Γ becomes

$$\boldsymbol{J}(\xi,\eta,0) := \begin{bmatrix} \dfrac{\partial x_h}{\partial \xi} & \dfrac{\partial y_h}{\partial \xi} & \dfrac{\partial z_h}{\partial \xi} \\ \dfrac{\partial x_h}{\partial \eta} & \dfrac{\partial y_h}{\partial \eta} & \dfrac{\partial z_h}{\partial \eta} \\ n_{h,x} & n_{h,y} & n_{h,z} \end{bmatrix}, \tag{1.30}$$

with

$$\frac{\partial x_h}{\partial \xi} = \sum_i \frac{\partial \psi_i(\xi,\eta)}{\partial \xi} x_{h,i}, \tag{1.31}$$

etc., and the normal is given by (1.25).

Using this methodology in combination with the tangential operator we have a way of computing the surface gradient of a function u

$$\nabla_\Gamma u := \left(\frac{\partial u_x}{\partial x_\Gamma}, \frac{\partial u_y}{\partial y_\Gamma}, \frac{\partial u_z}{\partial z_\Gamma} \right) =: \boldsymbol{P}\nabla u, \tag{1.32}$$

with the approximation

$$\nabla_\Gamma u \approx \sum_i \nabla_\Gamma \varphi_i U_i, \tag{1.33}$$

where U_i denotes the function value in node i of a mesh and

$$\nabla_\Gamma \varphi_i = \boldsymbol{P} \left(\frac{\partial \varphi_i}{\partial x}, \frac{\partial \varphi_i}{\partial y}, \frac{\partial \varphi_i}{\partial z} \right). \tag{1.34}$$

This approach has been used in Paper 3.

2 Trace Finite Element Method

CHAPTER INTRODUCTION

The *trace finite element method* (*TraceFEM*) makes it possible to discretize a surface independently of its description, effectively removing the need for fitting a mesh to the surface. Instead the surface is embedded into a fixed background mesh and allowed to arbitrarily intersect it. In this chapter an overview of the method is given. The original approach was suggested by Olshanskii et. al. in [44] where a new technique was introduced for hyper-surfaces.

A method closely related to TraceFEM is the *Cut Finite Element Method* (CutFEM) which incorporates ideas from the TraceFEM but is developed for solving cases involving interface problems with bulk interaction, see Burman et al. [6] for a review of CutFEM. In contrast to CutFEM, TraceFEM only deals with the integration over the interface domain, whereas CutFEM is also used to "cut" the finite element functions of the bulk domain at the interface and integrate over the resulting sub-domains on each side of the interface.

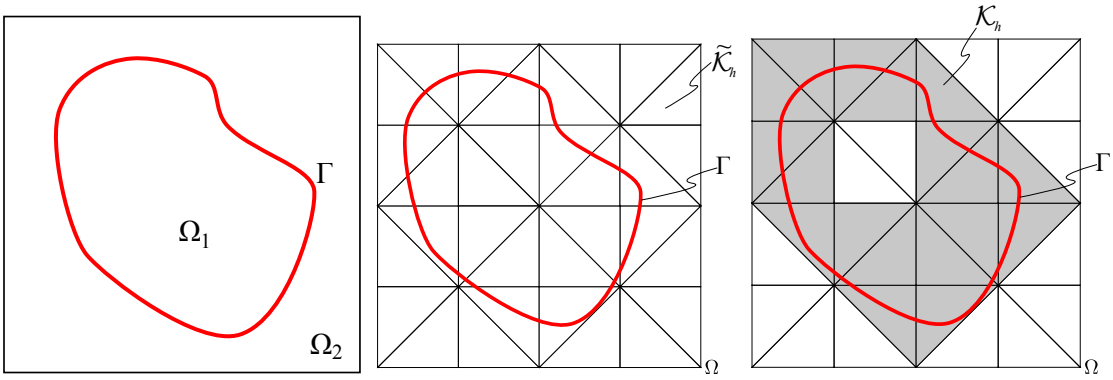
The basic idea of the TraceFEM/CutFEM is to let the surface Γ be embedded in a higher dimensional background mesh which is independent of the surface and is allowed to be intersected arbitrarily by Γ. The surface partial differential equation is then discretized by a set of background elements that are intersected by Γ and integrated over the approximation of the surface, i.e., using the *traces* of functions from the bulk finite element space on the approximation of the surface as described by Olshanskii and Reusken in [44], hence the term TraceFEM. This method is interesting as it avoids the explicit triangulation of the surface and is especially suitable for evolving surface problems such as the one in Paper 1, but it also allows for embedding arbitrarily shaped reinforcements to an elastic domain as in Paper 2.

Since the interface is intersecting the background mesh arbitrarily, some background elements might have very small cuts such that the element integrals will yield very small contributions to the resulting stiffness and mass matrices resulting in an ill-conditioned linear system. The TraceFEM/CutFEM approach thus needs stabilization, see Section 2.5 for details.

In order to find the locations of intersection between Γ and the background mesh, robust methods need to be established for 2D and 3D and for arbitrary polynomial order of a background element.

2.1 Domain

Let Γ denote a smooth d-dimensional interface contained by a higher dimensional domain Ω, which is a bounded sub-domain of $\mathbb{R}^{d+1}$. Furthermore, let Γ divide Ω into sub-domains such that $\Omega = \Omega_1 \cup \Omega_2 \cup \Gamma$. The interface can represent a surface in 3D or a curve in 2D, see Figure 2.1.



(a) (b) (c)

Figure 2.1: 2D representation of the problem domain

2.2 Discretization

The discretization is done by letting $\tilde{\mathcal{K}}_h$ denote the set of polyhedrons (3D) or polygons (2D) that tessellate the domain Ω completely with an element size parameter $0 < h < h_0$, this is known as the background or bulk mesh. The part of the background mesh used for the discretization of the problem is a subset of the whole mesh. We denote this as the *active* background mesh $\mathcal{K}_h \subseteq \tilde{\mathcal{K}}_h$ such that

$$\mathcal{K}_h = \{K \in \tilde{\mathcal{K}}_h : K \cap \Gamma \neq \emptyset\}, \tag{2.1}$$

see Figure 2.1.

Let $\phi : \mathbb{R}^d \to \mathbb{R}$ denote a continuous signed scalar level-set function such that,

$$\begin{aligned} \phi(\boldsymbol{x}) &< 0 \quad \text{in } \Omega_1, \\ \phi(\boldsymbol{x}) &= 0 \quad \text{on } \Gamma, \\ \phi(\boldsymbol{x}) &> 0 \quad \text{in } \Omega_2, \end{aligned}$$

ϕ is often chosen as the closest signed distance function to Γ but this is not a necessary property in general.

In practice ϕ might not be given as a continuous function of $\boldsymbol{x}$, for instance a level-set function

can be approximated from a CAD surface by computing e.g., the closest distance to the CAD surface in each background mesh node, yielding an approximation of a level-set function $\mathbf{\Phi}$ which is given at the nodes of the background mesh, see Figure 2.2.

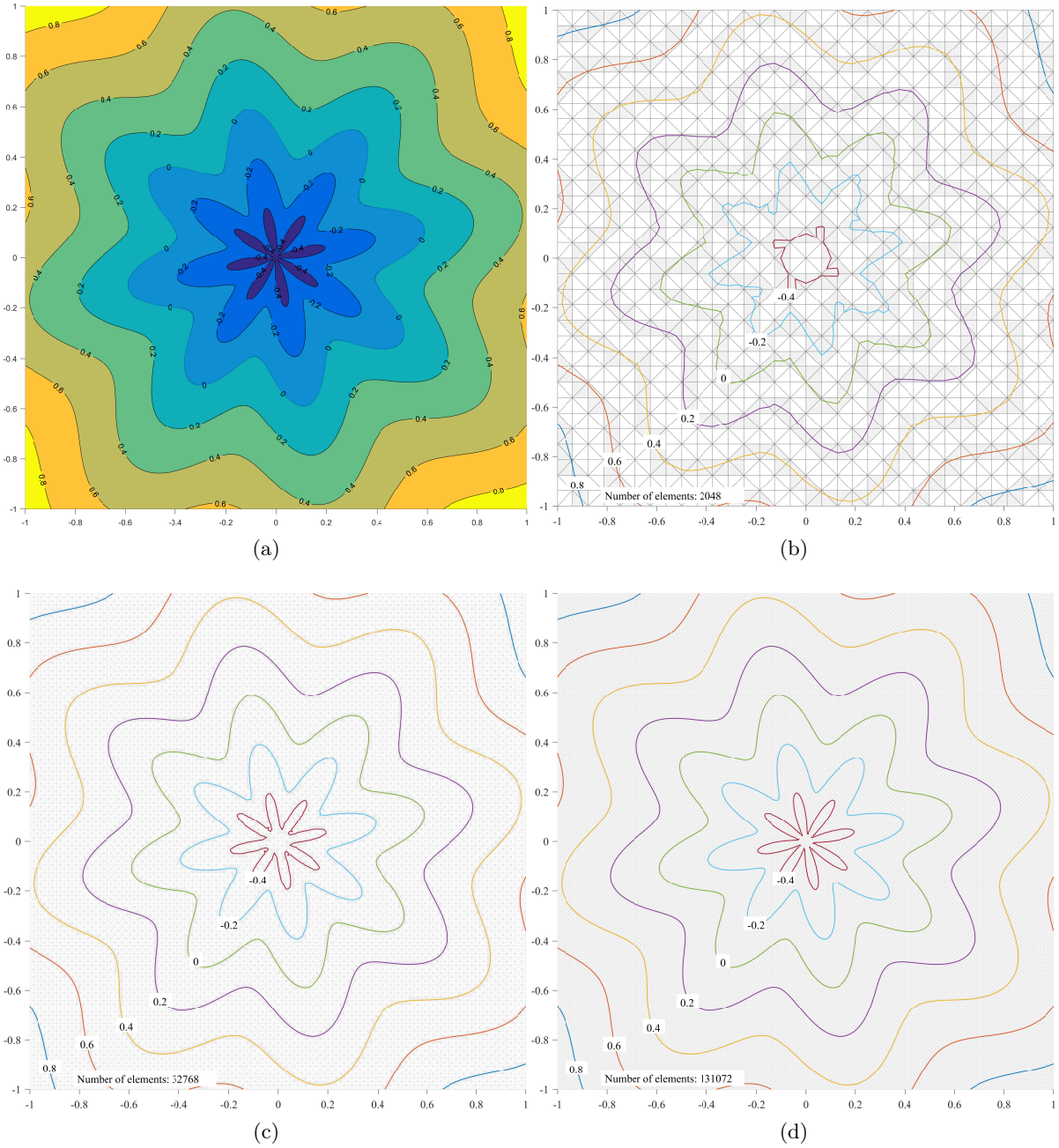


Figure 2.2: Difference between a continuous level-set function ϕ (a) and discrete $\mathbf{\Phi}$ (b)-(d). In (b)-(d) the level-set function is given at the nodes of the background mesh and the shown iso-contours are found by (in this case) linear interpolation. The level-set function is defined by $\phi = \sqrt{x^2 + y^2} - 0.5 + 0.1 \sin(8 \operatorname{atan}(y/x))$.

The (continuous) surface is given by the zero-level of the function ϕ

$$\Gamma = \{\boldsymbol{x} \in \Omega : \phi(\boldsymbol{x}) = 0\}. \tag{2.2}$$

We can interpolate $\boldsymbol{\Phi}$ by use of the basis functions of the bulk element K:

$$\phi_h(\boldsymbol{x}) = \sum_{i \in N_K} \varphi_i^{m_B}(\boldsymbol{x})\Phi_i, \tag{2.3}$$

where N_K is the set of nodes in K, Φ_i are the known nodal values of the signed distance function and $\varphi_i^{m_B}(\boldsymbol{x})$ is the basis function of polynomial order $m_B = \{1, 2\}$ acting on element K. The discrete zero-level set is then given by

$$\Gamma_h = \{\boldsymbol{x} \in \Omega : \Pi_h^{m_\Gamma}\phi_h(\boldsymbol{x}) = 0\}, \tag{2.4}$$

where the interpolant $\Pi_h^{m_\Gamma}$of polynomial order m_Γ depends on the bulk element type. Note that ϕ_h in (2.4) can be replaced with a continuous level-set function $\phi(\boldsymbol{x})$ to improve accuracy, see Section 2.6 for details. Examples of the resulting set of discrete zero-level sets is shown in Figure 2.3.

2.3 Dirichlet conditions

A simple way of dealing with Dirichlet boundary conditions in this setting is to directly prescribe the degrees of freedom on the boundary of the background mesh. Let $\partial\mathcal{K}_{h,D}$ denote the boundary of $\mathcal{K}_h$ that is intersected by the discrete surface boundary denoted as $\partial\Gamma_{h,D}$, see Figure 2.4. This straightforward approach was used in Paper 1 and 4. If however $\partial\Gamma_{h,D}$ is not intersecting $\partial\mathcal{K}_{h,D}$ then the background mesh must be carefully constructed to facilitate proper Dirichlet conditions, see Figure 2.4b. This was the situation in Paper 2 in the case of the oblate spheroid where the mesh needed modification since it was unstructured. For a more general handling of boundary conditions we could use Nitsche's method, as in [10, 9, 36]

2.4 Finite element space

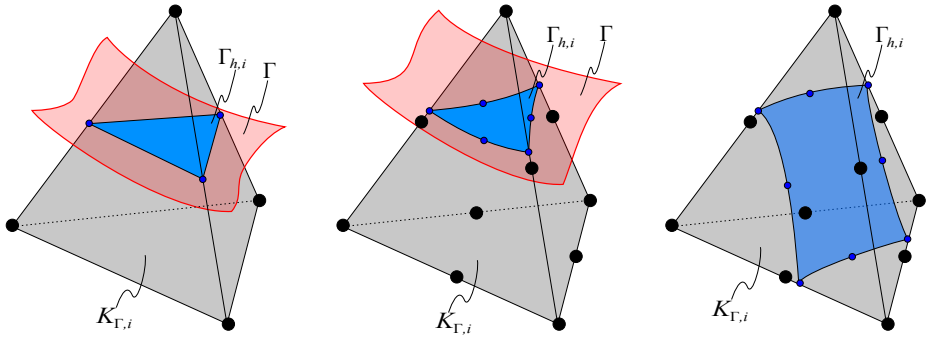
In the setting of the TraceFEM the finite element space is defined on the background mesh, and in particular when it is applied to surface problems, the finite element space is only defined on the active background mesh.

The finite element space is defined by

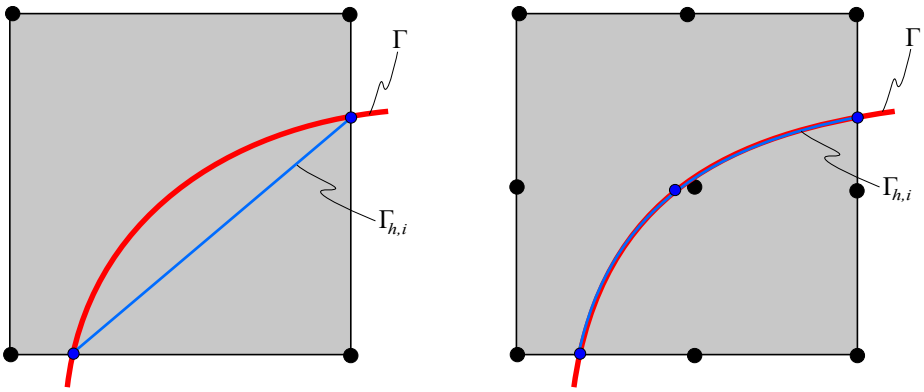
$$V_h = \left\{\boldsymbol{v} \in \tilde{V}_h^{m_B}|_{\mathcal{K}_h} : \boldsymbol{v} = \boldsymbol{0} \text{ on } \partial\mathcal{K}_{h,D}\right\} \tag{2.5}$$

where $\tilde{V}_h^{m_B}$ is a space of continuous piecewise polynomials of order $m_B = \{1, 2\}$ (subscript B denotes the bulk) defined on $\mathcal{K}_h$ and d denotes the dimension of the bulk element, typically $d = \{2, 3\}$.

Consider for example diffusion on a surface

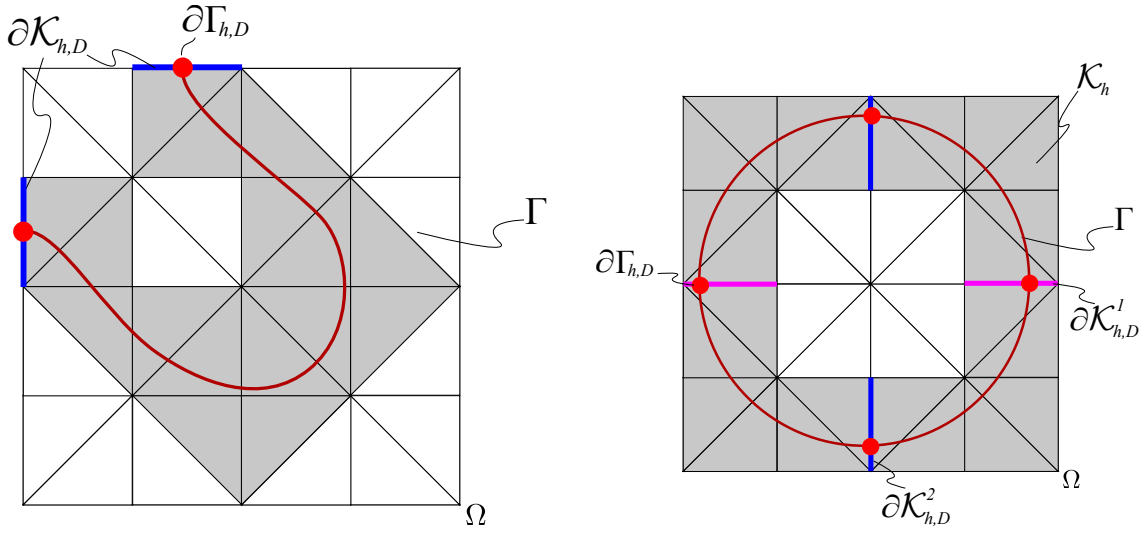


(a) 3D



(b) 2D

Figure 2.3: Discrete surface element types. (a) left: $\mathbf{\Phi}$ is linearly interpolated ($m_\Gamma = 1$) to yield a flat triangle element. (a) middle: quadratic interpolation ($m_\Gamma = 2$) yielding a second order triangle. (a) right: quadratic interpolation yielding a second order quadrilateral. (b) left: linear interpolation yielding a line element. (b) right: quadratic interpolation yielding a quadratic curve element.



(a) (b)

Figure 2.4: (a) Simple way of handling $\partial\Gamma_{h,D}$. (b) $\partial\mathcal{K}_{h,D} = \partial\mathcal{K}^1_{h,D} \cup \partial\mathcal{K}^2_{h,D}$ must be constructed carefully in case of an unstructured mesh in order to facilitate center lines, as shown in the figure, to ensure equilibrium in e.g., membrane problems with internal pressure.

$$-\nabla_\Gamma \cdot \nabla_\Gamma u = f \quad \text{on } \Gamma, \tag{2.6}$$

where $\nabla_\Gamma u$ is defined by 1.32
and

$$\nabla_\Gamma \cdot \boldsymbol{v} = \text{tr}(\nabla_\Gamma \otimes \boldsymbol{v}) \tag{2.7}$$

The finite element method on Γ_h is then given by: find the solution field $u_h \in V_h$ such that

$$a_h(u_h, v)_{\Gamma_h} = l_h(v)_{\Gamma_h} \quad \forall v \in V_h, \tag{2.8}$$

where

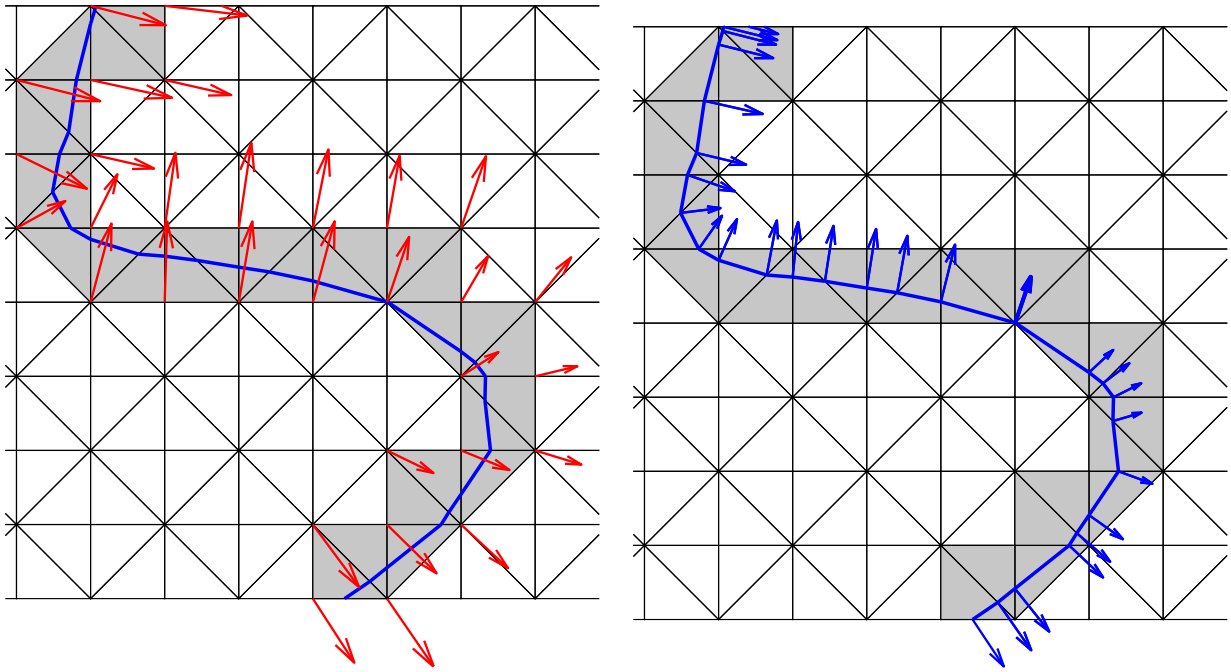
$$a_h(u_h, v)_{\Gamma_h} = \int_{\Gamma_h} \nabla_{\Gamma_h} u \cdot \nabla_{\Gamma_h} v d\Gamma_h \tag{2.9}$$

and

$$l_h(v)_{\Gamma_h} = \int_{\Gamma_h} f v d\Gamma_h \tag{2.10}$$

The solution u_h to (2.8) exists on the background mesh and in order to visualize the solution on Γ_h we need to interpolate the solution using shape functions of the background mesh

$$u_{\Gamma_h,j} = \sum \varphi_i(\boldsymbol{x}_\Gamma) u_{h,i} \tag{2.11}$$



(a) (b)

Figure 2.5: Discrete gradient field $\nabla\phi_h$ The blue line represents the linearly interpolated zero-level set Γ_h. (a) The red arrows represent the discrete gradient field $\nabla\phi_h$ projected on the "narrow band" $\mathcal{K}_h$. (b) The blue arrows represent the interpolated gradient field to the zero-level set.

see Figure 2.5.

2.5 Stabilization

Since (2.8) is integrated over the discrete interface Γ_h and because of the arbitrary cuts by the interface through the background mesh, there might be a large variation in the area of the resulting surface elements, see Figure 2.6. This can lead to a severely ill-conditioned stiffness and mass matrix since the condition number may become very large for certain cut cases. This is numerically shown in [9] where the condition number is plotted against some displacement of Γ, see Figure 2.7. Furthermore a finite element method of surface problems using higher dimensional shape functions can be unstable (see, e.g., [30], Papers 1, 2 and 4). Some finite element methods on surfaces can be unstable regardless whether TraceFEM or a standard triangulation of the surface is used, see e.g., Paper 3 and [30]. Recently, several stabilization methods for unfitted methods have been proposed in literature. These stabilization methods add terms to the stiffness or mass matrix in order to stabilize the method and/or to improve the conditioning. The method that is used in this thesis is the so called *ghost penalty* method originally proposed in [9]. Other methods exists, see, e.g., [43] for a recent overview. In this section we shall give a brief introduction to the method.

Note that for any element $K \in \mathcal{K}_h$ there exist at least one neighbor $K_N \in \mathcal{K}_h$ such that K and

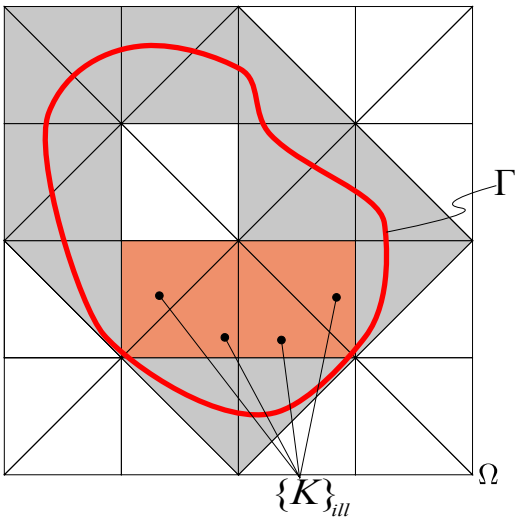


Figure 2.6: Elements $\{K\}_{ill}$ that cause ill-conditioning of the linear system.

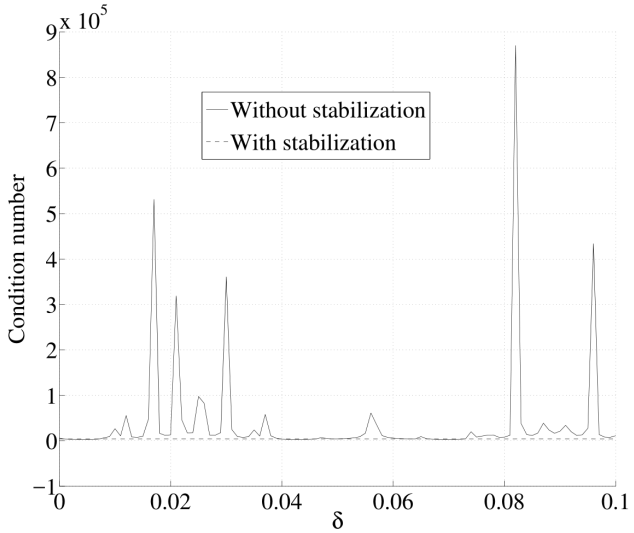
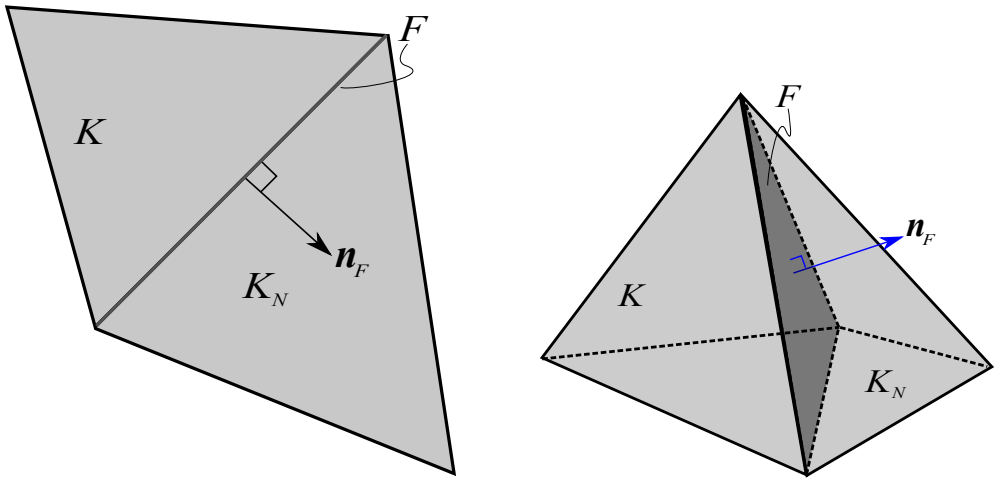


Figure 2.7: Condition number as a function of displacement δ of the interface Γ for the stabilized and unstabilized methods. (Adopted from [9] with the approval from the author)



(a) 2D case, where $\mathcal{F}$ is actually a set of edges.

(b) 3D case

Figure 2.8: Interior faces of the active elements, $\mathcal{K}_h$.

K_N share a face, see Figure 2.8. Define a set of interior faces of $\mathcal{K}_h$ by

$$\mathcal{F}_h = \{F = K \cap K_N : K, K_N \in \mathcal{K}_h\}. \tag{2.12}$$

Each internal face defines a unit normal vector $\boldsymbol{n}_F$ that is perpendicular to the face and oriented exterior to K , see Figure 2.8. The stabilization is constructed such that the jump of the gradient in the direction $\boldsymbol{n}_F$ across two background elements is penalized. The jump of the gradient of $\boldsymbol{v}$ across the face F of elements K and K_N is defined on each side of F as

$$[\boldsymbol{n}_F \cdot \nabla v] = \boldsymbol{n}_F \cdot \nabla v|_{K \bigcap F} - \boldsymbol{n}_F \cdot \nabla v|_{K_N \bigcap F}, \tag{2.13}$$

where $\boldsymbol{n}_F \cdot \nabla v|_{K \bigcap F}$ denotes the normal derivative evaluated at the face F of element K. The stabilized form of (2.8) becomes

$$a_h(v, w)_{\Gamma_h} + \gamma j_h(v, w) = l_h(v)_{\Gamma_h} \quad \forall v, w \in V_h. \tag{2.14}$$

The stabilizing term $j_h(\cdot, \cdot)$ is then given as the sum of all gradient jumps as

$$j_h(\boldsymbol{v}, \boldsymbol{w}) = \sum_{F \in \mathcal{F}_h} \int_F [\boldsymbol{n}_F \cdot \nabla \boldsymbol{v}] \cdot [\boldsymbol{n}_F \cdot \nabla \boldsymbol{w}] ds, \tag{2.15}$$

where γ is a positive scalar stabilization parameters that is user defined.

The stabilization term controls potential fluctuations of the solution field of the membrane problem in Paper 2 and 4 and can be seen as additional stiffness that constrains these fluctuations, see Figure 2.9 for a 2D example. If γ is chosen large enough the displacements will tend towards zero. Thus, there exists an optimal value of γ; for a discussion on the effect and choice of γ see Paper 3

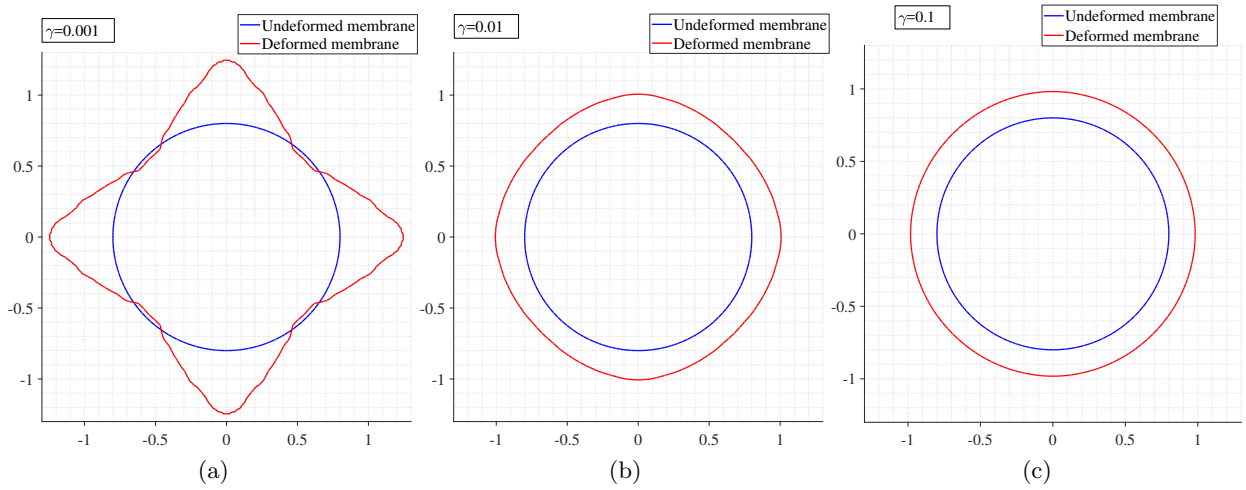


Figure 2.9: Effects of the stabilization parameter γ on a 2D membrane solution.

and 4 where it is shown that there exist a quite wide range near the optimal γ where the solution is stable. It can be concluded that the solution is fairly insensitive to the choice of γ. Implementation details can be found in Section 3.2.

2.6 Zero-level set reconstruction

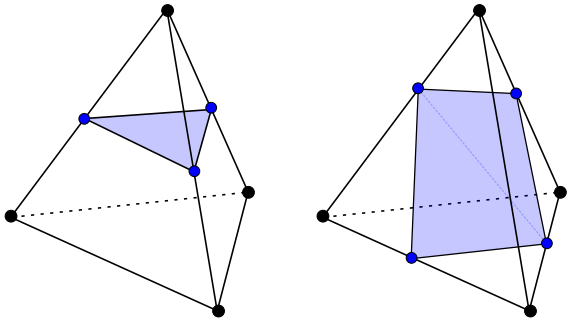
In this section we describe the approach for extracting the discrete zero-level set Γ_h from a signed distance function $\phi(\boldsymbol{x})$. Recall that the zero level set is defined by

$$\Gamma = \{\boldsymbol{x} \in \Omega : \phi(\boldsymbol{x}) = 0\}. \tag{2.16}$$

The basic idea is to determine the zero-level set for each element K by some form of root finding. The discrete zero-level set is given by using an interpolant that depends on the background mesh

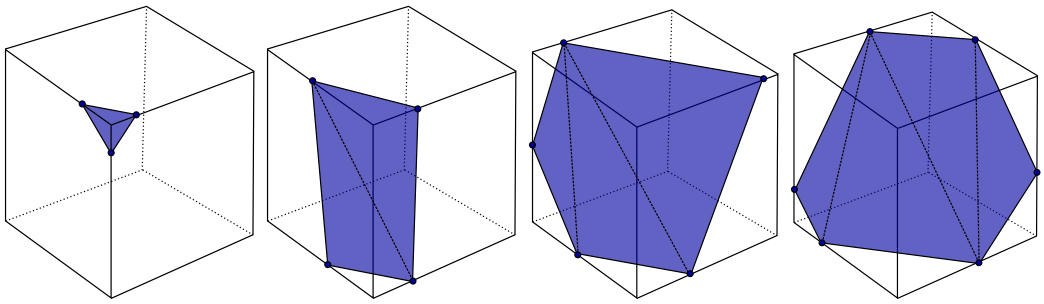
$$\Gamma_h = \{\boldsymbol{x} \in \Omega : \Pi_h^{m_\Gamma} \phi_h(\boldsymbol{x}) = 0\}, \tag{2.17}$$

which means that the roots to $\phi_h(\boldsymbol{x}) = 0$ are restricted to the faces of element K and that the number of surface points (roots) depend on the order m_Γ of the surface element, e.g., a P_1 tetrahedron could yield a P_1 triangle (3 points) or a P_1 quadrilateral (4 points). See Figure 2.10, 2.11 and 2.12 for various cut cases.



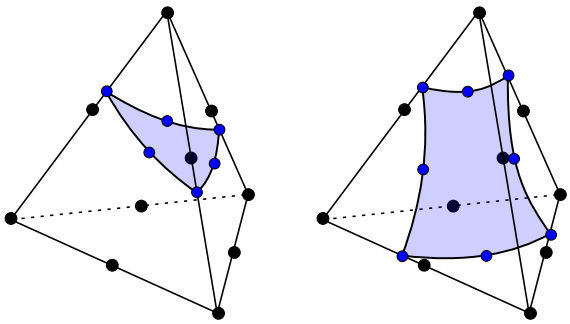
(a) Triangular (b) Quadrilateral

Figure 2.10: Linear tetrahedral cut cases.



(a) Triangular (b) Quadrilateral (c) Pentagon (d) Hexagon

Figure 2.11: Linear hexahedral cut cases.



(a) Triangular (b) Quadrilateral

Figure 2.12: Quadratic tetrahedral cut cases.

2.6.1 Valid topology

In order to create a robust method for the extraction of the zero-level set, we need to determine if the level set function is sufficiently resolved by the background mesh, which means that each cut element must yield a valid topology; see Figure 2.13 for examples of bad topology that can't be allowed. In case of a bad topology the corresponding background element is locally refined until the surface cut yields a valid topology, see, e.g, [25]. By valid topology we mean that the arbitrary

intersection of a zero-level set with the background element results in a number of surface points that can be mapped to surface polygons. To determine if an element $K \in \mathcal{K}_h$ is cut we follow the approach proposed by Fries et al. [25] and compute

$$\min_{i \in N_{\text{grid}}} \left(\Phi_i^{\text{grid}} \right) \cdot \max_{i \in N_{\text{grid}}} \left(\Phi_i^{\text{grid}} \right) < 0, \tag{2.18}$$

where

$$\Phi_j^{\text{grid}} = \sum_{i=1} \varphi_i^{m_B} \left(\boldsymbol{r}_j^{\text{grid}} \right) \cdot \Phi_i \quad \forall j \in N_{\text{grid}}, \tag{2.19}$$

$\boldsymbol{r}_j^{\text{grid}}$ denotes a number of uniformly spaced sample points in the parametric space (r, s, t). Note that $\varphi_i^{m_B} \left(\boldsymbol{r}_j^{\text{grid}} \right)$ can be computed in a pre-processing step, and re-used for every background element. In order to avoid numerical issues we make sure that no Φ_i is exactly zero by perturbing by some small number away from zero. To be certain that the surface topology is valid we check the following conditions on each face of the tetrahedral:

- Each edge of the face may only be cut once.
- The number of cuts per face must be two.
- If no face is cut, then all nodes of the tetrahedron must have the same sign and thus the whole tetrahedron is uncut.

In the case described in Figure 2.13d we identify high curvature by

$$\nabla \bar{\Phi}^{\text{grid}} \cdot \nabla \Phi_j^{\text{grid}} < tol, \tag{2.20}$$

where $\nabla \bar{\Phi}^{\text{grid}}$ denotes the average of all $\nabla \Phi_j^{\text{grid}}$ for all $j \in N_{\text{grid}}$ and tol is a user defined number chosen such that large differences in the angle between $\nabla \bar{\Phi}^{\text{grid}}$ and $\nabla \Phi_j^{\text{grid}}$ define high curvature, here $\nabla \Phi_j^{\text{grid}}$ is given by

$$\nabla \Phi_j^{\text{grid}} = \sum_{i=1} \nabla \varphi_i^{m_B} \left(\boldsymbol{r}_j^{\text{grid}} \right) \cdot \Phi_i. \tag{2.21}$$

2.6.2 Linear interpolant

For every face F of every element $K \in \mathcal{K}_h$ two cut points are determined along two edges by linear interpolation using the level-set function values and coordinates at the corners of F, see Figure 2.14. The following steps are used:

1. For every parent element $K_i \in \mathcal{K}_h$ loop over all edges e_i.
   a) For every edge e_i check the sign of the two discrete function values $\boldsymbol{\Phi}|_{e_i}$ to determine if the edge is cut.

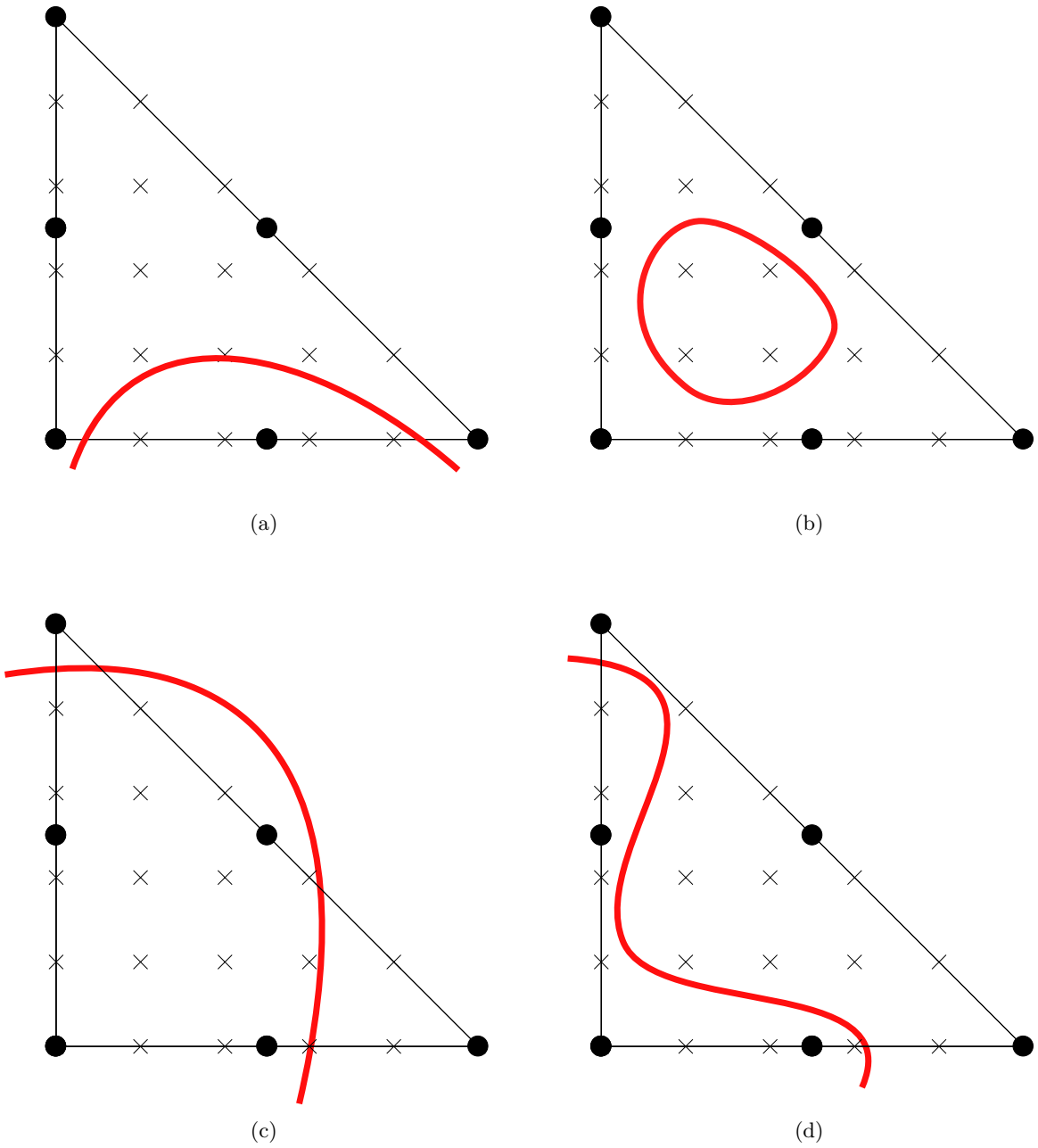


Figure 2.13: Examples of bad topologies visualized on side-views of a 3D parametric second order tetrahedral element with uniformly distributed sample points. The red curves represent a surface. a) One edge cut more than once. b) A small interface is completely enclosed by an element. c) More than two edges are cut. d) The curvature of the cut is too high.

b) Linearly interpolate the cut point $\mathbf{x}_{\Gamma,i}$ along the edge e_i using the two vertex coordinates $\mathbf{x}^k_{e_i}$and $\mathbf{x}^l_{e_i}$, at nodes k and l (endpoints of e_i) and the function values $\mathbf{\Phi}|_{e_i} = \{\phi(\mathbf{x}^k_{e_i}), \phi(\mathbf{x}^l_{e_i})\}$.

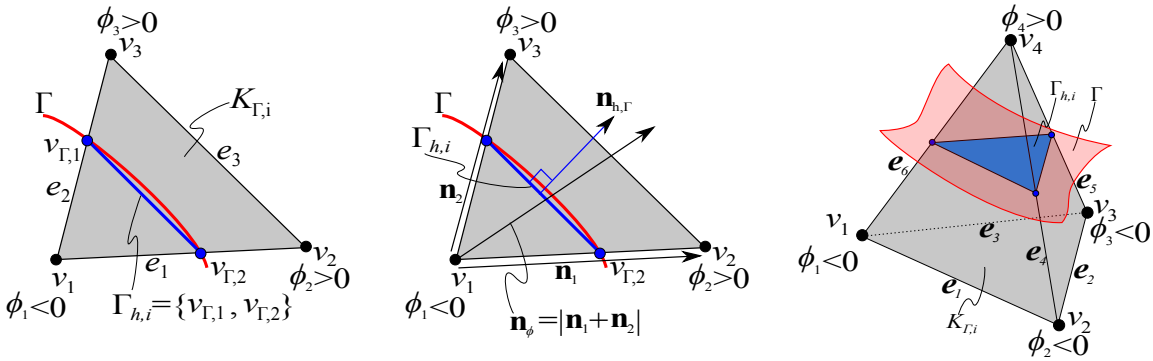
c) Let $\mathbf{x}_{e_i,1} = \mathbf{x}_{e_i}|_{\phi|_{e_i}>0}$ (the coordinate corresponding to the highest value of ϕ) and $\mathbf{x}_{e_i,0} = \mathbf{x}_{e_i}|_{\phi|_{e_i}<0}$ and compute the vector $\mathbf{n}_i = \mathbf{x}_{e_i,1} - \mathbf{x}_{e_i,0}$. See figure 2.14b.

2. In order to determine the orientation of the face normals, compute the element vector $\mathbf{n}_\phi = \sum \mathbf{n}_i$ which is pointing in the general direction of $\nabla\phi$.

3. Depending on the number of nodes in element K^Γ_i and the orientation of the cut, several cut cases must be considered, see Figure 2.10 for tetrahedral element and figure 2.11 for hexahedra.

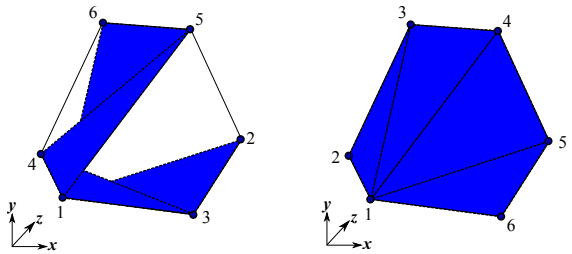
4. The resulting polygon is tessellated into triangles by rotating the arbitrary polygon from $\mathbb{R}^3$ into $\mathbb{R}^2$ and applying a convex hull algorithm, see Figure 2.15.

5. The complete tessellation of all triangles on Γ_h is then processed by a triangulation algorithm to create a compact topology list which is used for the linear interpolation of the solution on $\mathcal{K}_h$ to Γ_h (see Section 3.7 for a discussion on the topology) and for visualization (easier handling and interpolated surface shading).



(a) 2D case (b) Surface element normal (c) 3D case

Figure 2.14: Surface element Γ^i_h and parent element K^i_Γ in 2D and 3D.



(a) Wrong numbering (b) Proper numbering order

Figure 2.15: Numbering order for arbitrary polygon in $\mathbb{R}^3$. The order before renumbering is shown in (a). In (b) the polygon is rotated into $\mathbb{R}^2$ and using an convex hull algorithm the proper numbering order can be established for any convex polygon.

2.6.3 Higher order interpolant

In the case of linear interpolation it was sufficient to find the roots along the edges of an element using a direct linear interpolation method. If the background elements are of a higher order $m_B > 1$ we need to find additional roots along the faces as well as edges and the problem becomes less trivial. In order to create a robust method we need to utilize iterative root-finding algorithms such as Newton's method. Using an iterative method means that we need access to $\phi(\boldsymbol{x})$, but in general, a continuous $\phi(\boldsymbol{x})$ might not be known, instead we may only have access to a discrete signed distance function $\boldsymbol{\Phi}$ defined in the nodes of K. In this case we can create an approximation of ϕusing of the basis functions of the bulk element K:

$$\phi_h(\boldsymbol{x}) = \sum_{i \in N_K} \varphi_i^{m_B}(\boldsymbol{x}) \phi_{h,i}, \tag{2.22}$$

where N_K is the set of nodes in K, Φ_i are the known nodal values of the signed distance function and $\varphi_i^{m_B}(\boldsymbol{x})$ is the basis function of polynomial order $m_B > 1$ acting on element K. Note that the basis functions can alternatively be mapped or defined in the physical coordinate system since the bulk element is assumed affine.

In order to find the roots for $\phi_h(\boldsymbol{x}) = 0$, when $\phi_h(\boldsymbol{x})$ is interpolated using an interpolant Π_h^m of order $m > 1$, we follow the work done in [25, 24] using the following steps:

1. For each element check if the topology is valid by following the steps in the previous Section 2.6.1.

2. For each face F of tetrahedral element K in $\mathcal{K}_h|_m$ the nodal values of $\phi_h|_F$ are mapped to a parametric triangle $T^{\mathrm{r}}|_m$, see Figure 2.16. If element K has a valid topology it's faces must have either two or zero cut edges, additionally at least three faces must be cut. We determine if the face is cut and which edges are cut by following a procedure analogues to (2.18). Additionally we renumber the nodes of the faces such that they are unique, i.e., the normal of each face $F_{K_i \cap K_j}$, no matter which tetrahedral element they belong to, points in

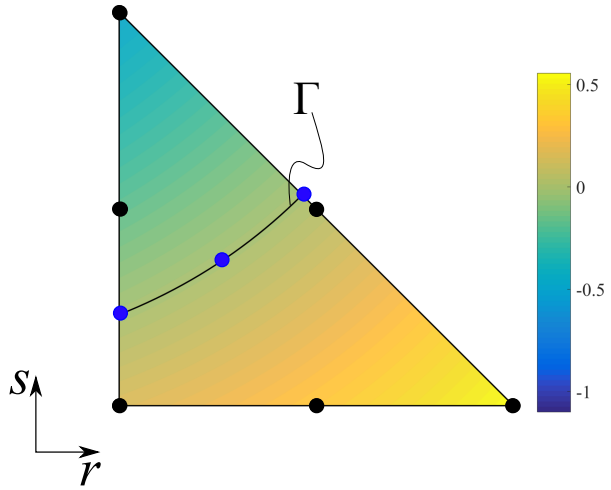


Figure 2.16: Isocontours of the distance function ϕ on a mapped face of a tetrahedral element.

the same direction, $\boldsymbol{n}|_{F_{K_i \cap K_j}} = \boldsymbol{n}|_{F_{K_j \cap K_i}}$. This ensures that the gradients computed with the isoparametric map on the face elements are the same for both elements K_i and K_j, otherwise the edge-points of the reconstructed surface elements might not coincide, see Figure 2.18. The resulting surface is thus guarantied to be C_0 continuous.

3. On each cut edge on the parametric face T_m^r we employ a Newton-Raphson iterative search scheme:

$$\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \frac{\phi_h(\boldsymbol{r}_i)}{\nabla\phi_h(\boldsymbol{r}_i) \cdot \boldsymbol{s}}\boldsymbol{s}, \tag{2.23}$$

where $\boldsymbol{r} = [r, s]$ is the local coordinate of the parametric triangle, $\nabla\phi_h(\boldsymbol{r}_i)$ is evaluated by interpolation using the basis functions and $\boldsymbol{s}$ is the search direction. To find the root along the edges, $\boldsymbol{s}$ is simply the directional vector along the edge.

4. Once the two edge points are found the inner node needs to be determined by the same root finding scheme. It turns out that the search direction is critical for the convergence of the Newton search as well as the geometrical convergence as shown in [25, 24], where the authors propose 5 different variations of the search directions and 2 ways of starting position of the search. Choosing a linearly interpolated starting position (straight line between the edge roots) and set the search direction to be the normal to the line or $\boldsymbol{s} = \nabla\phi_h(\boldsymbol{r}_0)$ yields satisfying results with respect to accuracy and performance, see [24]. In some rare cases when the Newton search fails if gets stuck in a false root lying outside of the triangle, in this case we employ bisection in order to get back inside the triangle where the Newton search is continued until convergence. This approach yields a robust method in all cases but increases the number of iterations slightly for these rare cases.

5. In order to orient the surface, the resulting surface points need to be numbered such that

their normal is oriented in the same general direction as $\nabla\bar{\Phi}^{\text{grid}}$.

6. The resulting surface elements from a tetrahedral element are either triangular or quadrilateral and in the later case can be split into triangular elements, albeit with additional root finding for the mid-node. It is possible to use other types of surface elements such as the serendipity element, see Paper 4. We denote the resulting surface topology set by $\mathcal{T}$.

7. If the resulting discrete surface needs to be used for smooth surface shading, then an additional step is needed to create a connectivity from the list of unconnected surface elements. In order to accomplish this efficiently the background mesh information for each surface patch is used to uniquely number the nodes and create the connectivity list. Note that this step is not necessary for integration.

If we have access to the exact function ϕ, the procedure above is still valid, with the difference that we need to map $\boldsymbol{r}_i$ to $\boldsymbol{x}$ before evaluating $\phi(\boldsymbol{x}(\boldsymbol{r}_i))$ and $\nabla\phi(\boldsymbol{x}(\boldsymbol{r}_i))$.

In case the background elements are affine, it is possible to create the above scheme in physical coordinates by evaluating the basis functions in physical coordinates, $\varphi(\boldsymbol{x})$, see Section 3.8. The search for roots on edges in this case is the same as above, the search on faces however is "free" since $\boldsymbol{s} = \nabla\phi(\boldsymbol{x}_0)$. In this case we restrict $\boldsymbol{s}$ to the (planar) face of the tetrahedron by tangential projection:

$$\boldsymbol{s}_F := \boldsymbol{P}_F \nabla\phi(\boldsymbol{x}_0), \tag{2.24}$$

where $\boldsymbol{P}_F = \boldsymbol{I} - \boldsymbol{n}_F \otimes \boldsymbol{n}_F$, $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{n}_F \otimes \boldsymbol{n}_F$ the outer product of the face normal to the tetrahedron face, see Figure 2.17. Note that the construction of $\boldsymbol{\varphi}(\boldsymbol{x})$ is done to avoid the mapping of $\boldsymbol{r}$ to $\boldsymbol{x}$ in each step of the root finding algorithm.

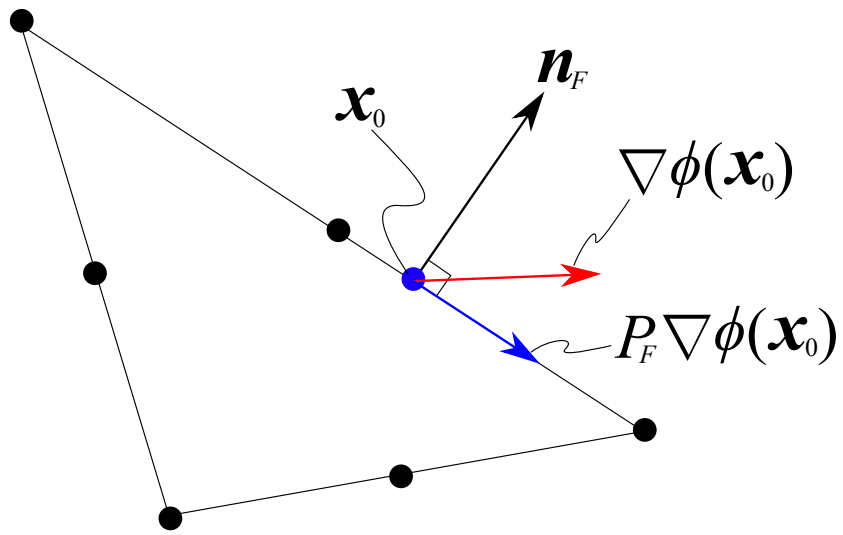


Figure 2.17: Side view of a tetrahedral element. The search direction $\nabla\phi$ is projected onto the tetrahedral face F (shown here as a line) resulting in a modified Newton method with the search direction $P_F\nabla\phi$.

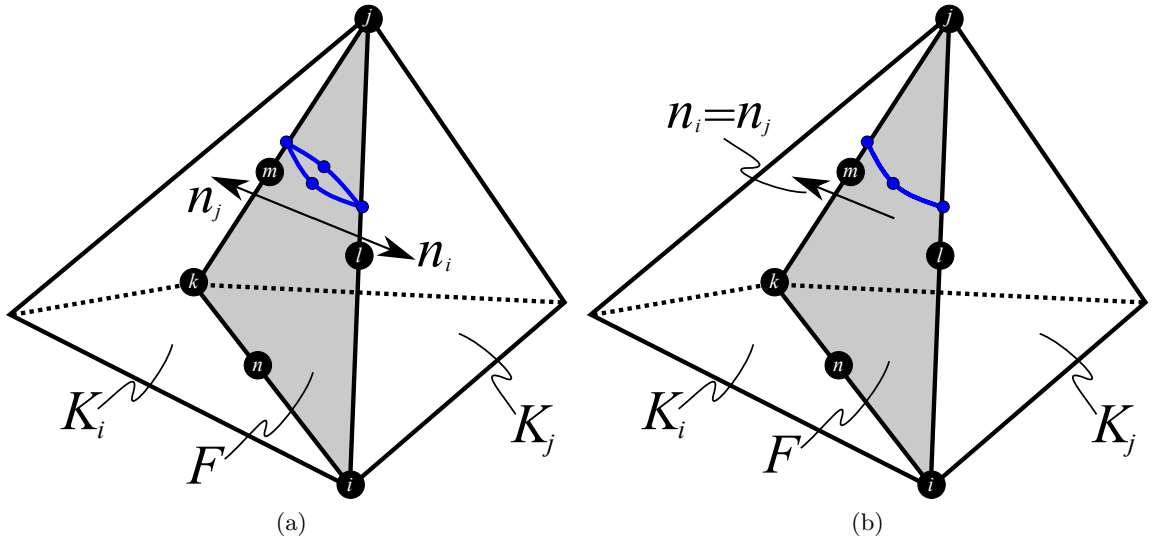


Figure 2.18: Face numbering. (a) $F_{K_i} = \{i, j, k, l, m, n\}$, $F_{K_j} = \{i, k, j, n, m, l\}$. (b) $F_{K_i} = F_{K_j} = \{i, j, k, l, m, n\}$.

3 Implementation Details

3.1 Choice of software

Prototypes of the algorithms throughout this thesis have been implemented in MATLAB which allows for vectorization of the code in order to improve performance. Note that the main focus was generating scripts to test a concept and therefore little time was spent on the performance analysis and optimization of the implementation. There is thus much room for optimizing the performance of the algorithms and implementations. The main reason for the lack of performance optimization and complexity analysis is that the code was written to be as readable as possible in order to minimize implementation error. We have to create *working* code before we can focus on optimizing it for performance. High level languages offer faster implementation times with a trade-off in performance. Furthermore, MATLAB, while getting increasingly better at its JIT compiler[1] is still an interpreted language with its primary use in generating prototypes for testing and creating proofs of concept. Performance can however be improved through some extra effort, especially in the assembly process, where an indexed approach [15] is utilized which is particularly interesting as it allows for parallelization of the assembly process to considerably improve performance in, e.g., iterative algorithms.

3.2 Stabilization

For vector valued unknowns $\boldsymbol{u}$ we have $\boldsymbol{u} \approx \boldsymbol{\Phi}\mathbf{u}$, where $\mathbf{u}$ denotes nodal values and

$$\boldsymbol{\Phi} := \begin{bmatrix} \varphi_1 & 0 & 0 & \varphi_2 & 0 & 0 & \cdots \\ 0 & \varphi_1 & 0 & 0 & \varphi_2 & 0 & \cdots \\ 0 & 0 & \varphi_1 & 0 & 0 & \varphi_2 & \cdots \end{bmatrix}, \tag{3.1}$$

and with $\boldsymbol{n}_h$ denoting the discrete face normal of a given face, we define

[1] http://mathworks.com/products/matlab/matlab-execution-engine/

$$\mathbf{B}_F := [(\boldsymbol{n}_h \cdot \nabla)\boldsymbol{\Phi}, -(\boldsymbol{n}_h \cdot \nabla)\boldsymbol{\Phi}] = \begin{bmatrix} \left(\sum_i n_{h,x_i} \dfrac{\partial}{\partial x_i}\right) \varphi_1 & \cdots \\ \vdots & \ddots \end{bmatrix}, \tag{3.2}$$

and thus the discrete stabilization matrix is given by

$$\mathbf{J} = \sum_{F \in \mathcal{F}} \int_F \mathbf{B}_F^\intercal \mathbf{B}_F dF. \tag{3.3}$$

In the scalar case of (2.14), we instead use $\boldsymbol{\Phi}_s := [\varphi_1, \varphi_1, \cdots]$ and $\mathbf{B}_{F,s} := [(\boldsymbol{n}_h \cdot \nabla)\boldsymbol{\Phi}_s, -(\boldsymbol{n}_h \cdot \nabla)\boldsymbol{\Phi}_s]$ such that the stabilized linear system of (2.14) takes the form

$$(\mathbf{S} + \gamma \mathbf{J})\, \mathbf{u} = \mathbf{f}. \tag{3.4}$$

3.3 Curvature flow

The minimal surface problem is a classical example of a certain type of partial differential equations on surfaces called *form finding*. This section gives an overview of the approach taken in Paper 1 to compute the curvature of a surface and iteratively find the design that minimizes it.

We let $\Gamma(t)$ denote a time dependent surface and consider the minimal surface problem: Given a final time T, find $\boldsymbol{x}_\Gamma : \Gamma(t) \to \mathbb{R}^3$ such that

$$\dot{\boldsymbol{x}}_\Gamma = \Delta_\Gamma \boldsymbol{x}_\Gamma = -2H\boldsymbol{n} \quad \text{in } \Gamma(t),\ t \in (0, T) \tag{3.5}$$

Here,

$$\dot{\boldsymbol{x}}_\Gamma := \frac{\partial \boldsymbol{x}_\Gamma}{\partial t}, \tag{3.6}$$

Δ_Γ denotes the Laplace-Beltrami operator defined by

$$\Delta_\Gamma = \nabla_\Gamma \cdot \nabla_\Gamma, \tag{3.7}$$

where ∇_Γ is the surface gradient given by (1.32). H is the mean curvature of $\Gamma(t)$

$$H = \frac{\kappa_1 + \kappa_2}{2}, \tag{3.8}$$

where $\kappa_1$and κ_2 are the principal curvatures see e.g. [3].

If we let a zero level to a level set function ϕ coincide with $\boldsymbol{x}_\Gamma$, we can utilize the material derivative of ϕ

$$\frac{D\phi}{Dt} = \frac{\partial \phi}{\partial t} + \dot{\boldsymbol{x}}_\Gamma \cdot \nabla \phi = 0 \tag{3.9}$$

The surface is deformed by letting the level set be evolved by advection, see e.g., [47, 46] for more information on the level set method. If $|\nabla\phi| = 1$ then $\nabla\phi|_\Gamma = \boldsymbol{n}$ and we have

$$\frac{\partial\phi}{\partial t} = -\boldsymbol{u}_\Gamma \cdot \boldsymbol{n}, \tag{3.10}$$

where $\boldsymbol{u}_\Gamma = \dot{\boldsymbol{x}}_\Gamma$. In order to find $\boldsymbol{x}_\Gamma$ for some time t we first compute $\boldsymbol{u}_\Gamma = \Delta_\Gamma \boldsymbol{x}_\Gamma$ and then update ϕ by (3.10). This process is iterated until the final time T is reached, which happens when the curvature of Γ is below some threshold. The advantage of this method is that the surface is treated as an implicit surface and is not bound by the computational requirements of an explicit surface. The surface is instead free to evolve through a domain and can naturally be split and merged without complicated meshing techniques.

For the normal curvature flow from (3.5) the bilinear form is given by the problem: Given Γ_h^n and the corresponding coordinate function $\boldsymbol{x}_{\Gamma,h}^n$ at a time step n, find $\boldsymbol{u}_{\Gamma,h}^n \in [V_h]^3$ such that

$$\left(\boldsymbol{u}_{\Gamma,h}^n, \boldsymbol{v}\right)_{\Gamma_h^n} + j(\boldsymbol{u}_{\Gamma,h}^n, \boldsymbol{v}) = -a\left(\boldsymbol{x}_{\Gamma,h}^n, \boldsymbol{v}\right)_{\Gamma_h^n} \quad \forall \boldsymbol{v} \in [V_h]^3, \tag{3.11}$$

where

$$V_h = \{\boldsymbol{v} \in \text{ piecewise linear polynomial defined on the background mesh , } \boldsymbol{v} = \boldsymbol{0} \text{ on } \partial\Gamma\}, \tag{3.12}$$

$j(\boldsymbol{u}_{\Gamma,h}^n, \boldsymbol{v})$ is a stabilization term as described in Section (3.2), $a\left(\boldsymbol{x}_{\Gamma,h}^n, \boldsymbol{v}\right)_{\Gamma_h^n} = \int_{\Gamma_h^n} \nabla \boldsymbol{x}_\Gamma \cdot \nabla \boldsymbol{v} d\Gamma_h^n$ and $(\boldsymbol{u}, \boldsymbol{v})_\Gamma = \int_\Gamma \boldsymbol{u} \cdot \boldsymbol{v} d\Gamma$ is the L_2 inner product on Γ. We thus have

$$\int_\Gamma \boldsymbol{u}_{\Gamma,h}^n \cdot \boldsymbol{v} d\Gamma + \gamma \mathbf{J} = -\int_{\Gamma_h^n} \nabla \boldsymbol{x}_{\Gamma,h}^n \cdot \nabla \boldsymbol{v} d\Gamma_h^n \tag{3.13}$$

which leads to the discrete system

$$(\mathbf{M} + \gamma \mathbf{J})\, \mathbf{u}_n = -\mathbf{S}\mathbf{x}_n, \tag{3.14}$$

where $\mathbf{M}$ is the mass matrix,

$$\mathbf{M} = \int_\Gamma \mathbf{\Phi}^\intercal \mathbf{\Phi} d\Gamma, \tag{3.15}$$

and $\mathbf{\Phi}$ is given by (3.1) with φ denoting the basis functions of V_h and $\mathbf{S}$ is the stiffness matrix given on Voigt form by

$$\mathbf{S} = \int_\Gamma \mathbf{B}_\Gamma^\intercal \mathbf{B}_\Gamma d\Gamma, \tag{3.16}$$

where

$$\mathbf{B}_\Gamma = \begin{bmatrix} \varphi^1_{\Gamma,x} & 0 & 0 & \varphi^2_{\Gamma,x} & 0 & 0 & 0 & \dots \\ \varphi^1_{\Gamma,y} & 0 & 0 & \varphi^2_{\Gamma,y} & 0 & 0 & 0 & \dots \\ \varphi^1_{\Gamma,z} & 0 & 0 & \varphi^2_{\Gamma,z} & 0 & 0 & 0 & \dots \\ 0 & \varphi^1_{\Gamma,x} & 0 & 0 & \varphi^2_{\Gamma,x} & 0 & 0 & \dots \\ 0 & \varphi^1_{\Gamma,y} & 0 & 0 & \varphi^2_{\Gamma,y} & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{3.17}$$

and

$$\varphi^1_{\Gamma,x} = \frac{\partial \varphi_1}{\partial x_\Gamma}, \tag{3.18}$$

etc. since we have

$$\nabla_\Gamma \boldsymbol{\varphi} = \boldsymbol{P}_\Gamma \nabla \boldsymbol{\varphi}, \tag{3.19}$$

with

$$\nabla \boldsymbol{\varphi} = \begin{bmatrix} \varphi^1_x & \varphi^2_x & \dots \\ \varphi^1_y & \varphi^2_y & \dots \\ \varphi^1_z & \varphi^2_z & \dots \end{bmatrix}, \tag{3.20}$$

and $\varphi^1_x = \dfrac{\partial \varphi_1}{\partial x}$, etc.

3.4 Level set advection

Using $\mathbf{u}_n$ from (3.14) we choose a time step k_n and propagate ϕ^n_h to ϕ^{n+1}_h on the narrow band of background elements $\mathcal{K}_{h,N}$ (see Figure (3.1)) around Γ using (3.10)

$$\phi^{n+1}_h = \phi^{n+1}_h - k_n \mathbf{u}_n \cdot \boldsymbol{n}^n_h, \tag{3.21}$$

where $\boldsymbol{n}^n_h$ is the $L_2(\Gamma^n)$-projection of the discontinuous normal vector obtained from the discrete level set function

$$\mathbf{n}_{\text{disc}} = \frac{\nabla \phi^n_h}{|\nabla \phi^n_h|}. \tag{3.22}$$

The normal $\boldsymbol{n}^n_h \in V^n_h$ is then computed from

$$(\boldsymbol{n}^n_h, \boldsymbol{v})_{\Gamma^n_h} = (\mathbf{n}_{\text{disc}}, \boldsymbol{v})_{\Gamma^n_h} \quad \forall \boldsymbol{v} \in V^n_h. \tag{3.23}$$

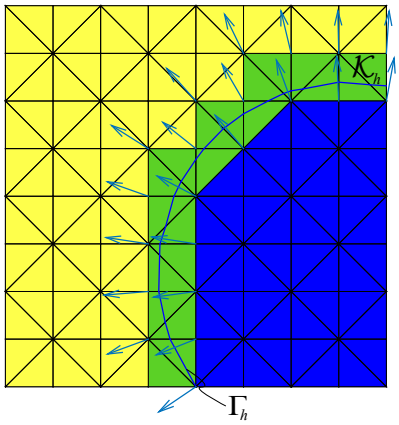


Figure 3.1: Discontinuous gradient field on $\mathcal{K}_h$

It is assumed that $\nabla\phi = \boldsymbol{n}$ at $\phi = 0$, so it must hold that $|\nabla\phi| = 1$ i.e. ϕ must be a signed distance function for all virtual time steps n. After computing ϕ_{n+1} using (3.10) the property $|\nabla\phi| = 1$ gets degraded and the distance function needs to be reinitialized. This can be done in various ways e.g. [48, 47] the naive approach is to compute the closest distance from each node on the narrow band of the mesh to the discrete interface.

3.5 The L_2-projection

Given some function $\boldsymbol{u} \in [L_2(\Omega)]^3$ the L_2-projection $\boldsymbol{u}_h \in [V_h]^3$ of $\boldsymbol{u}$ is defined by

$$\int_\Omega (\boldsymbol{u} - \boldsymbol{u}_h)\, \boldsymbol{v} d\Omega = 0, \ \forall \boldsymbol{v} \in [V_h]^3, \tag{3.24}$$

i.e., the L_2-projection takes an arbitrary function $\boldsymbol{u}$ into the finite element space V_h by minimizing the L_2 norm of $(\boldsymbol{u} - \boldsymbol{u}_h)$ such that

$$(\boldsymbol{u}, \boldsymbol{v})_\Omega = (\boldsymbol{u}_h, \boldsymbol{v})_\Omega \,. \tag{3.25}$$

See Figure (3.2). Or equivalently

$$\mathbf{M}\mathbf{u}_N = \mathbf{f}, \tag{3.26}$$

where

$$\mathbf{f} = \int_\Omega \mathbf{\Phi}^\intercal \mathbf{U} d\Omega, \tag{3.27}$$

with $\mathbf{U}$ denoting the element value for $\boldsymbol{u}$ and where $\mathbf{\Phi}$ is given by (3.1) and

$$\mathbf{M} = \int_\Omega \mathbf{\Phi}^\intercal \mathbf{\Phi} d\Omega. \tag{3.28}$$

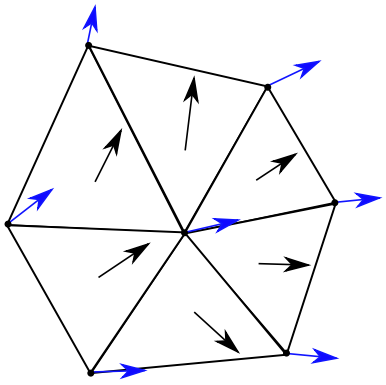


Figure 3.2: L_2-projection for averaging the element values onto the nodal values.

3.6 Linear elastic membrane model

In order to describe the implementation of the linear elastic membrane model, we recapitulate the equations from Paper 2.

The surface strain tensor is defined by

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) := \frac{1}{2}\left(\nabla_\Gamma \otimes \boldsymbol{u} + (\nabla_\Gamma \otimes \boldsymbol{u})^\mathsf{T}\right) \tag{3.29}$$

and the in-plane strain tensor is given by:

$$\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u}) := \boldsymbol{\varepsilon}(\boldsymbol{u}) - ((\boldsymbol{\varepsilon}(\boldsymbol{u}) \cdot \boldsymbol{n}) \otimes \boldsymbol{n} + \boldsymbol{n} \otimes (\boldsymbol{\varepsilon}(\boldsymbol{u}))) \tag{3.30}$$

The model problem is then to find $\boldsymbol{u} : \Gamma \to \mathbb{R}^3$ and $\boldsymbol{\sigma}_\Gamma : \Gamma \to \mathbb{R}^3$ such that

$$-\nabla_\Gamma \cdot \boldsymbol{\sigma}_\Gamma = \boldsymbol{f} \quad \text{on} \quad \Gamma \tag{3.31}$$

$$\boldsymbol{\sigma}_\Gamma = 2\mu\boldsymbol{\varepsilon}_\Gamma + \lambda_0 \mathrm{tr}(\boldsymbol{\varepsilon}_\Gamma)\boldsymbol{P}_\Gamma \quad \text{on} \quad \Gamma \tag{3.32}$$

$$\boldsymbol{u} = \boldsymbol{0} \quad \text{on} \quad \partial\Gamma_D \tag{3.33}$$

where $\partial\Gamma_D$ are Dirichlet conditions and $\boldsymbol{f} : \Gamma \to \mathbb{R}^3$ is a load per unit area. The Lamé coefficients are given by

$$\lambda_0 := \frac{2\lambda\mu}{\lambda + 2\mu} = \frac{E\nu}{1-\nu^2},\ \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)},\ \mu = \frac{E}{2(1+\nu)} \tag{3.34}$$

where E is the Young's modulus and ν is the Poisson's ratio. λ_0 is used in the plane stress case when small thickness is assumed.

The bilinear form of 3.31 is

$$a_\Gamma(\boldsymbol{u}, \boldsymbol{v}) = l_\Gamma(\boldsymbol{v}) \quad \forall \boldsymbol{v} \in V_h, \tag{3.35}$$

where

$$a_\Gamma(\boldsymbol{u}, \boldsymbol{v}) = (2\mu\boldsymbol{\varepsilon}(\boldsymbol{u}), \boldsymbol{\varepsilon}(\boldsymbol{v}))_\Gamma - (4\mu\boldsymbol{\varepsilon}(\boldsymbol{u}) \cdot \boldsymbol{n}, \boldsymbol{\varepsilon}(\boldsymbol{v}) \cdot \boldsymbol{n})_\Gamma + (\lambda_0 \nabla_\Gamma \cdot \boldsymbol{u}, \nabla_\Gamma \cdot \boldsymbol{v})_\Gamma \,, \tag{3.36}$$

and

$$l_\Gamma(\boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v})_\Gamma. \tag{3.37}$$

Which is equivalent to

$$\begin{aligned} &\int_\Gamma 2\mu\boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v}) d\Gamma - \int_\Gamma 4\mu(\boldsymbol{\varepsilon}(\boldsymbol{u}) \cdot \boldsymbol{n}) \cdot (\boldsymbol{\varepsilon}(\boldsymbol{v}) \cdot \boldsymbol{n}) d\Gamma \\ &+ \int_\Gamma \lambda_0 (\nabla_\Gamma \cdot \boldsymbol{u}) \cdot (\nabla_\Gamma \cdot \boldsymbol{v}) d\Gamma = \int_\Gamma \boldsymbol{f}\boldsymbol{v} d\Gamma \end{aligned} \tag{3.38}$$

Using Mandel notation [31] we have the strain tensor

$$\boldsymbol{\varepsilon}_\Gamma^M := \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \sqrt{2}\varepsilon_{12} \\ \sqrt{2}\varepsilon_{13} \\ \sqrt{2}\varepsilon_{23} \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial x_\Gamma} & 0 & 0 \\ 0 & \frac{\partial}{\partial y_\Gamma} & 0 \\ 0 & 0 & \frac{\partial}{\partial z_\Gamma} \\ \frac{1}{\sqrt{2}}\frac{\partial}{\partial x_\Gamma} & \frac{1}{\sqrt{2}}\frac{\partial}{\partial y_\Gamma} & 0 \\ \frac{1}{\sqrt{2}}\frac{\partial}{\partial x_\Gamma} & 0 & \frac{1}{\sqrt{2}}\frac{\partial}{\partial z_\Gamma} \\ 0 & \frac{1}{\sqrt{2}}\frac{\partial}{\partial y_\Gamma} & \frac{1}{\sqrt{2}}\frac{\partial}{\partial z_\Gamma} \end{pmatrix} \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}, \tag{3.39}$$

and using

$$\boldsymbol{\Phi} := \begin{pmatrix} \varphi_1 & 0 & 0 & \varphi_2 & 0 & 0 & \cdots \\ 0 & \varphi_1 & 0 & 0 & \varphi_2 & 0 & \cdots \\ 0 & 0 & \varphi_2 & 0 & 0 & \varphi_2 & \cdots \end{pmatrix}, \tag{3.40}$$

we get

$$\mathbf{B}_\varepsilon := \begin{pmatrix} \frac{\partial}{\partial x_\Gamma} & 0 & 0 \\ 0 & \frac{\partial}{\partial y_\Gamma} & 0 \\ 0 & 0 & \frac{\partial}{\partial z_\Gamma} \\ \frac{1}{\sqrt{2}}\frac{\partial}{\partial x_\Gamma} & \frac{1}{\sqrt{2}}\frac{\partial}{\partial y_\Gamma} & 0 \\ \frac{1}{\sqrt{2}}\frac{\partial}{\partial x_\Gamma} & 0 & \frac{1}{\sqrt{2}}\frac{\partial}{\partial z_\Gamma} \\ 0 & \frac{1}{\sqrt{2}}\frac{\partial}{\partial y_\Gamma} & \frac{1}{\sqrt{2}}\frac{\partial}{\partial z_\Gamma} \end{pmatrix} \boldsymbol{\Phi} \tag{3.41}$$

such that the Galerkin approximation of $\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u}) : \boldsymbol{\varepsilon}_\Gamma(\boldsymbol{v})$ yields $\mathbf{B}_\varepsilon^\intercal \mathbf{B}_\varepsilon \mathbf{U}$ and for the first term of

(3.38)

$$\int_\Gamma 2\mu\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u}) : \boldsymbol{\varepsilon}_\Gamma(\boldsymbol{v})d\Gamma \approx \left(\int_\Gamma 2\mu\mathbf{B}_\varepsilon^\intercal\mathbf{B}_\varepsilon d\Gamma\right)\mathbf{U}. \tag{3.42}$$

For the second term of (3.38) we have that

$$\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u})\cdot\boldsymbol{n} = \begin{pmatrix} n_1\frac{\partial u_x}{\partial x_\Gamma} + n_2\frac{1}{2}\left(\frac{\partial u_y}{\partial x_\Gamma}+\frac{\partial u_x}{\partial y_\Gamma}\right) + n_3\frac{1}{2}\left(\frac{\partial u_z}{\partial x_\Gamma}+\frac{\partial u_x}{\partial z_\Gamma}\right) \\ n_1\frac{1}{2}\left(\frac{\partial u_y}{\partial x_\Gamma}+\frac{\partial u_x}{\partial y_\Gamma}\right) + n_2\frac{\partial u_y}{\partial y_\Gamma} + n_3\frac{1}{2}\left(\frac{\partial u_y}{\partial z_\Gamma}+\frac{\partial u_z}{\partial y_\Gamma}\right) \\ n_1\frac{1}{2}\left(\frac{\partial u_z}{\partial x_\Gamma}+\frac{\partial u_x}{\partial z_\Gamma}\right) + n_2\frac{1}{2}\left(\frac{\partial u_y}{\partial z_\Gamma}+\frac{\partial u_z}{\partial y_\Gamma}\right) + n_3\frac{\partial u_z}{\partial z_\Gamma} \end{pmatrix}, \tag{3.43}$$

introducing the notation system

$$\boldsymbol{\varphi}^1_{x_\Gamma} := \begin{pmatrix}\frac{\partial\varphi_1}{\partial x_\Gamma} & 0 & 0 & \frac{\partial\varphi_2}{\partial x_\Gamma} & 0 & 0 & \cdots\end{pmatrix}, \tag{3.44}$$

$$\boldsymbol{\varphi}^2_{x_\Gamma} := \begin{pmatrix}0 & \frac{\partial\varphi_1}{\partial x_\Gamma} & 0 & 0 & \frac{\partial\varphi_2}{\partial x_\Gamma} & 0 & \cdots\end{pmatrix}, \tag{3.45}$$

$$\boldsymbol{\varphi}^3_{x_\Gamma} := \begin{pmatrix}0 & 0 & \frac{\partial\varphi_1}{\partial x_\Gamma} & 0 & 0 & \frac{\partial\varphi_2}{\partial x_\Gamma} & \cdots\end{pmatrix}, \tag{3.46}$$

$$\boldsymbol{\varphi}^1_{y_\Gamma} := \begin{pmatrix}\frac{\partial\varphi_1}{\partial y_\Gamma} & 0 & 0 & \frac{\partial\varphi_2}{\partial y_\Gamma} & 0 & 0 & \cdots\end{pmatrix}, \tag{3.47}$$

etc., we have

$$\mathbf{B}_n = \begin{pmatrix} n_1\boldsymbol{\varphi}^1_{x_\Gamma} + n_2\frac{1}{2}\left(\boldsymbol{\varphi}^1_{y_\Gamma}+\boldsymbol{\varphi}^2_{x_\Gamma}\right) + n_3\frac{1}{2}\left(\boldsymbol{\varphi}^1_{z_\Gamma}+\boldsymbol{\varphi}^3_{x_\Gamma}\right) \\ n_1\frac{1}{2}\left(\boldsymbol{\varphi}^1_{y_\Gamma}+\boldsymbol{\varphi}^2_{x_\Gamma}\right) + n_2\boldsymbol{\varphi}^2_{y} + n_3\frac{1}{2}\left(\boldsymbol{\varphi}^2_{z_\Gamma}+\boldsymbol{\varphi}^3_{y_\Gamma}\right) \\ n_1\frac{1}{2}\left(\boldsymbol{\varphi}^1_{z_\Gamma}+\boldsymbol{\varphi}^3_{x_\Gamma}\right) + n_2\frac{1}{2}\left(\boldsymbol{\varphi}^2_{z_\Gamma}+\boldsymbol{\varphi}^3_{y_\Gamma}\right) + n_3\boldsymbol{\varphi}^3_{z_\Gamma} \end{pmatrix} \tag{3.48}$$

and the second term approximation becomes

$$\int_\Gamma 4\mu(\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u})\cdot\boldsymbol{n})\cdot(\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{v})\cdot\boldsymbol{n})d\Gamma \approx \left(\int_\Gamma 4\mu\mathbf{B}_n^\intercal\mathbf{B}_n d\Gamma\right)\mathbf{U}. \tag{3.49}$$

For the third term we use the notation

$$\mathbf{B}_{div} := \begin{pmatrix}\frac{\partial\varphi_1}{\partial x_\Gamma} & \frac{\partial\varphi_1}{\partial y_\Gamma} & \frac{\partial\varphi_1}{\partial z_\Gamma} & \frac{\partial\varphi_2}{\partial x_\Gamma} & \frac{\partial\varphi_2}{\partial y_\Gamma} & \frac{\partial\varphi_2}{\partial z_\Gamma} & \cdots\end{pmatrix}, \tag{3.50}$$

and get

$$\int_\Gamma \lambda_0(\nabla_\Gamma\cdot\boldsymbol{u})\cdot(\nabla_\Gamma\cdot\boldsymbol{v})d\Gamma \approx \left(\int_\Gamma \lambda_0\mathbf{B}_{\text{div}}^\intercal\mathbf{B}_{\text{div}}d\Gamma\right)\mathbf{U}. \tag{3.51}$$

Finally we have the linear system

$$\left(\int_\Gamma 2\mu\mathbf{B}_\varepsilon^\intercal\mathbf{B}_\varepsilon d\Gamma - \int_\Gamma 4\mu\mathbf{B}_n^\intercal\mathbf{B}_n d\Gamma + \int_\Gamma \lambda_0\mathbf{B}_{\text{div}}^\intercal\mathbf{B}_{\text{div}}d\Gamma\right)\mathbf{U} = \int_\Gamma \boldsymbol{f}\boldsymbol{\Phi}d\Gamma \tag{3.52}$$

or

$$\mathbf{SU} = \mathbf{F} \tag{3.53}$$

3.7 Interpolating solution field to surface

Because small deformations are assumed the solution field of $\boldsymbol{u}_h|_{\mathcal{K}_h}$ can be interpolated to the surface $\boldsymbol{u}_h|_{\Gamma_h}$ by

$$\mathbf{u}_{\Gamma,K} = \boldsymbol{\varphi}_K \cdot \mathbf{u}_K, \tag{3.54}$$

for each element K, where $\mathbf{u}_{\Gamma,K}$ denotes the solution field on Γ_h, $\mathbf{u}_K$ denotes the solution field of the background element and $\boldsymbol{\varphi}_K$ is the basis function of element K.

3.8 Evaluation of basis functions in physical coordinates

In order to construct $\varphi(\boldsymbol{x})$ on an affine second order tetrahedron we define the geometric interpolation using the sub-parametric mapping

$$\boldsymbol{x} = \sum_{i=1}^{4} \tilde{\varphi}_i \boldsymbol{x}_i \tag{3.55}$$

where $\tilde{\varphi}_i$ are the linear basis function on the corner nodes of a 10-noded tetrahedral element, with numbering according to Figure 3.3, and $\boldsymbol{x}_i$ are the corresponding coordinates. We expand 3.55 and get

$$(1 - r - s - t)\boldsymbol{x}_1 + r\boldsymbol{x}_2 + s\boldsymbol{x}_3 + t\boldsymbol{x}_4 = \boldsymbol{x} \tag{3.56}$$

which on matrix form is

$$\mathbf{A}\boldsymbol{r} + \boldsymbol{x}_1 = \boldsymbol{x} \tag{3.57}$$

where

$$\mathbf{A} = \begin{bmatrix} \boldsymbol{x}_2 - \boldsymbol{x}_1 & \boldsymbol{x}_3 - \boldsymbol{x}_1 & \boldsymbol{x}_4 - \boldsymbol{x}_1 \end{bmatrix} \tag{3.58}$$

with $\boldsymbol{x}_i = [x_i, y_i, z_i]^\intercal$. We solve for $\boldsymbol{r}$ and get

$$\boldsymbol{r}(\boldsymbol{x}) = \mathbf{A}^{-1}(\boldsymbol{x} - \boldsymbol{x}_1). \tag{3.59}$$

Using the full basis function for the 10-noded tetrahedron φ evaluated at $\boldsymbol{r}(\boldsymbol{x})$ we can write

$$\varphi(\boldsymbol{x}) = \varphi(\boldsymbol{r}(\boldsymbol{x})) \tag{3.60}$$

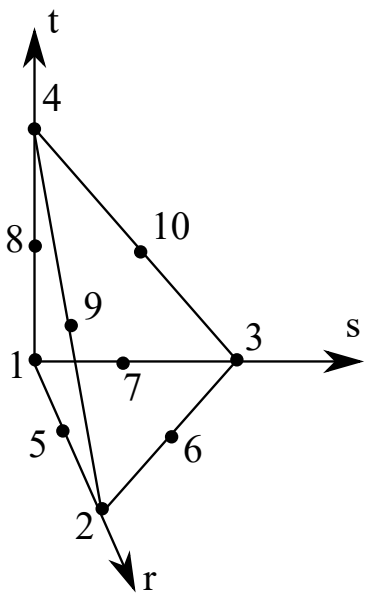


Figure 3.3: Tetrahedral node numbering.

and analogously

$$\nabla\varphi(\boldsymbol{x}) = \nabla\varphi(\boldsymbol{r}(\boldsymbol{x})). \tag{3.61}$$

Note that for every background element K, $\mathbf{A}^{-1}$ needs only be computed once, which improves the performence of the root finding method.

4 Future Work

During this work, several possibilities for future studies have been identified.

Mechanics

Current 3D structural optimization challenges include the need for manual post-processing of a solution, which might be mitigated by employing a level set based CutFEM scheme. This would mean that the boundary of a topology is represented as a surface that is moved by the advection of the level-set. In each time step of the optimization, CutFEM is used to compute side conditions such as, e.g., stresses. The optimized topology is then no longer a jagged object that needs manual work to be used in subsequent analysis, but a triangulation that can (with some smoothing) be utilized directly.

Another idea is to create anisotropic material models by embedding thin one dimensional fibers into a three dimensional bulk to add stiffness. This is interesting for injection molding applications where current FE software can simulate an object that is injection molded and output a configuration of discrete 1D fibers. This output could be used as input data to create computational anisotropic material models using an approach similar to the one in Paper 2.

Sandwich constructions is another topic that can be explored. Glued laminated timber for instance can be modeled as elastic bulk material with embedded surfaces of different material. The coupling between the elements can be modeled with discontinuous Galerkin methods which makes it possible to introduce weakening and crack propagation in the models.

Computer graphics

The Gaussian curvature problem on flat triangles might be tackled by using an accurate mesh refinement technique in order to interpolate a flat triangle into 4 which can be mapped onto a P2 element such that the Weingarten map can be approximated accurately from which the Gaussian (and principle) curvatures are computed accurately.

TraceFEM technology

In order to create robust computational methods in both TraceFEM and CutFEM that can be used on practical problems we need to address some areas of the TraceFEM.

Adaptivity could be used to resolve the underlying mesh where the curvature of a surface is high.

The challenge is to develop a local refinement technique for the background mesh and to interpolate the discrete level-set function on the new grid-points accurately. Here, the approach by Fries et al. [24] to interpolate the level-set function on new grid points could be used.

Sharp features is another area which needs attention if we want to do any practical modeling with TraceFEM/ CutFEM. Using several level sets it is possible to create implicit surfaces with sharp features and in combination with Nitsche's method a FEM could be constructed.

Since the tri-linear hexahedral TraceFEM generated good results and a smooth solution in Paper 2, a higher order hexahedral element approach might be worth further investigation. The challenges include dealing with a large set of cut cases and the consequent numbering of the resulting curved and possible concave polygon.

Bibliography

[1] J. A. Bærentzen, J. Gravesen, F. Anton, and H. Aanæs. *Guide to computational geometry processing: foundations, algorithms, and methods*. Springer Science & Business Media, 2012.

[2] T. Belytschko, C. Parimi, N. Moës, N. Sukumar, and S. Usui. Structured extended finite element methods for solids defined by implicit surfaces. *International journal for numerical methods in engineering*, 56(4):609–635, 2003.

[3] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy. *Polygon mesh processing*. CRC press, 2010.

[4] E. Burman. Ghost penalty. *C. R. Math. Acad. Sci. Paris*, 348(21-22):1217–1220, 2010.

[5] E. Burman, S. Claus, P. Hansbo, M. G. Larson, and A. Massing. Cutfem: Discretizing geometry and partial differential equations. *International Journal for Numerical Methods in Engineering*, 104(7):472–501, 2015.

[6] E. Burman, S. Claus, P. Hansbo, M. G. Larson, and A. Massing. Cutfem: Discretizing geometry and partial differential equations. *International Journal for Numerical Methods in Engineering*, 104(7):472–501, 2015.

[7] E. Burman and P. Hansbo. Fictitious domain finite element methods using cut elements: I. A stabilized Lagrange multiplier method. *Computer Methods in Applied Mechanics and Engineering*, 199(41-44):2680–2686, 2010.

[8] E. Burman and P. Hansbo. Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method. *Applied Numerical Mathematics*, 62(4):328–341, 2012.

[9] E. Burman, P. Hansbo, and M. G. Larson. A stabilized cut finite element method for partial differential equations on surfaces: The laplace–beltrami operator. *Computer Methods in Applied Mechanics and Engineering*, 285:188–207, 2015.

[10] E. Burman, P. Hansbo, M. G. Larson, K. Larsson, and A. Massing. Finite element approximation of the laplace-beltrami operator on a surface with boundary. *arXiv preprint arXiv:1509.08597*, 2015.

[11] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76. ACM, 2001.

[12] D. Chapelle and K.-J. Bathe. *The finite element analysis of shells-Fundamentals*. Springer Science & Business Media, 2010.

[13] P. G. Ciarlet. Mathematical modelling of linearly elastic shells. *Acta Numerica 2001*, 10:103–214, 2001.

[14] K. Crane, C. Weischedel, and M. Wardetzky. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG)*, 32(5):152, 2013.

[15] F. Cuvelier, C. Japhet, and G. Scarella. An efficient way to perform the assembly of finite element matrices in matlab and octave. *arXiv preprint arXiv:1305.3122*, 2013.

[16] M. Delfour and J. Zolésio. A boundary differential equation for thin shells. *Journal of differential equations*, 119(2):426–449, 1995.

[17] M. Delfour and J. Zolésio. Tangential differential equations for dynamical thin/shallow shells. *Journal of differential equations*, 128(1):125–167, 1996.

[18] M. Delfour and J. Zolésio. Shapes and geometries: Metrics. *Analysis, Differential Calculus, and Optimization, SIAM, Philadelphia*, 2011.

[19] M. C. Delfour and J.-P. Zolésio. Differential equations for linear shells: comparison between intrinsic and classical models. *Advances in mathematical sciences: CRMs*, 25:41–124, 1997.

[20] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 317–324, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[21] G. Dziuk. Finite elements for the beltrami operator on arbitrary surfaces. In *Partial differential equations and calculus of variations*, pages 142–155. Springer, 1988.

[22] G. Dziuk and C. M. Elliott. Finite element methods for surface pdes. *Acta Numerica*, 22:289–396, 2013.

[23] O. Frei and B. Rasch. Finding form: towards an architecture of the minimal. *Axel Menges, Stuttgard*, 1995.

[24] T. Fries, S. Omerović, D. Schöllhammer, and J. Steidl. Higher-order meshing of implicit geometries - part: I integration and interpolation in cut elements. *Computer Methods in Applied Mechanics and Engineering*, 313:759–784, 2017.

[25] T.-P. Fries and S. Omerović. Higher-order accurate integration of implicit geometries. *International Journal for Numerical Methods in Engineering*, 2015.

[26] H. Gouraud. Continuous shading of curved surfaces. *IEEE transactions on computers*, 100(6):623–629, 1971.

[27] M. E. Gurtin and A. Ian Murdoch. A continuum theory of elastic material surfaces. *Archive for Rational Mechanics and Analysis*, 57(4):291–323, 1975.

[28] P. Hansbo, M. Larson, and F. Larsson. Tangential differential calculus and the finite element modeling of a large deformation elastic membrane problem. *Computational Mechanics*, 56:87–95, 2015.

[29] P. Hansbo and M. G. Larson. Finite element modeling of a linear membrane shell problem using tangential differential calculus. *Computer Methods in Applied Mechanics and Engineering*, 270:1–14, 2014.

[30] P. Hansbo, M. G. Larson, and S. Zahedi. Stabilized finite element approximation of the mean curvature vector on closed surfaces. *SIAM Journal on Numerical Analysis*, 53(4):1806–1832, 2015.

[31] P. Helnwein. Some remarks on the compressed matrix representation of symmetric second-order and fourth-order tensors. *Computer methods in applied mechanics and engineering*, 190(22):2753–2770, 2001.

[32] G. A. Holzapfel. *Nonlinear solid mechanics*, volume 24. Wiley Chichester, 2000.

[33] J. F. Hughes, A. Van Dam, J. D. Foley, and S. K. Feiner. *Computer graphics: principles and practice*. Pearson Education, 2014.

[34] S. Jin, R. R. Lewis, and D. West. A comparison of algorithms for vertex normal computation. *The Visual Computer*, 21(1-2):71–82, 2005.

[35] M. Joulaian, S. Hubrich, and A. Düster. Numerical integration of discontinuities on arbitrary domains based on moment fitting. *Computational Mechanics*, 57(6):979–999, 2016.

[36] C. Lehrenfeld. A higher order isoparametric fictitious domain method for level set domains. *arXiv preprint arXiv:1612.02561*, 2016.

[37] E. Magid, O. Soldea, and E. Rivlin. A comparison of gaussian and mean curvature estimation methods on triangular meshes of range image data. *Computer Vision and Image Understanding*, 107(3):139–159, 2007.

[38] N. Max. Weights for computing vertex normals from facet normals. *Journal of Graphics Tools*, 4(2):1–6, 1999.

[39] D. S. Meek and D. J. Walton. On surface normal and gaussian curvature approximations given data sampled from a smooth surface. *Computer Aided Geometric Design*, 17(6):521–543, 2000.

[40] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*, Math. Vis., pages 35–57. Springer, Berlin, 2003.

[41] M. Moumnassi, S. Belouettar, É. Béchet, S. P. Bordas, D. Quoirin, and M. Potier-Ferry. Finite element analysis on implicitly defined domains: An accurate representation based on arbitrary parametric surfaces. *Computer Methods in Applied Mechanics and Engineering*, 200(5):774–796, 2011.

[42] B. Müller, F. Kummer, and M. Oberlack. Highly accurate surface and volume integration on implicit domains by means of moment-fitting. *International Journal for Numerical Methods in Engineering*, 96(8):512–528, 2013.

[43] M. A. Olshanskii and A. Reusken. Trace finite element methods for pdes on surfaces. *arXiv preprint arXiv:1612.00054*, 2016.

[44] M. A. Olshanskii, A. Reusken, and J. Grande. A finite element method for elliptic equations on surfaces. *SIAM journal on numerical analysis*, 47(5):3339–3358, 2009.

[45] M. A. Olshanskii and D. Safin. Numerical integration over implicitly defined domains for higher order unfitted finite element methods. *Lobachevskii Journal of Mathematics*, 37(5):582–596, 2016.

[46] S. Osher and R. Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer Science & Business Media, 2006.

[47] J. A. Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press, 1999.

[48] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational physics*, 114(1):146–159, 1994.

[49] G. Thürrner and C. A. Wüthrich. Computing vertex normals from polygonal facets. *Journal of Graphics Tools*, 3(1):43–46, 1998.

[50] O. C. Zienkiewicz. *The finite element method in engineering science*. McGraw-Hill London, second edition, 1971.