LICENTIATE THESIS

Finite element methods on surfaces

Mirza Cenanovic

JÖNKÖPING UNIVERSITY
School of Engineering

Department of Product Development
SCHOOL OF ENGINEERING, JÖNKÖPING UNIVERSITY
Jönköping, Sweden 2015

Finite element methods on surfaces

Mirza Cenanovic

Simulation and Optimization
Department of Product Development
School of Engineering, Jönköping University
SE-551 11 Jönköping, Sweden
Mirza.Cenanovic@ju.se

Research Series from the School of Engineering, Jönköping University
Department of Product Development
Dissertation Series No 012, 2015
ISBN 978-91-87289-13-2

Prepared with LyX and LaTeX

Published and Distributed by
School of Engineering, Jönköping University
Department of Product Development
SE-551 11 Jönköping, Sweden

Printed in Sweden by
Ineko AB
Kållered, 2015

Abstract

The purpose of this thesis is to improve numerical simulations of surface problems. Two novel computational concepts are analyzed and applied on two surface problems; minimal surface problems and elastic membrane problems. The concept of tangential projection implies that direct computation on the surface is made possible compared to the classical approach of mapping 2D parametric surfaces to 3D surfaces by means of differential geometry operators. The second concept presented is the *cut finite element method*, in which the basic idea of discretization is to embed the $d-1$-dimensional surface in a d-dimensional mesh and use the basis functions of a higher dimensional mesh but integrate over the surface. The aim of this thesis is to present the basics of the two main approaches and to provide details on the implementation.

Supplements

The following supplements constitute the basis of this thesis.

Supplement I Mirza Cennanovic, Peter Hansbo, Mats G Larsson
Minimal surface computation using finite element method on an embedded surface

Cenanovic did the numerical implementations. Hansbo and Larsson supplied the theoretical framework.

Supplement II Mirza Cennanovic, Peter Hansbo, Mats G Larsson
Cut finite element modeling of linear membranes

Cenanovic did the numerical implementations. Hansbo and Larsson supplied the theoretical framework.

Supplement III Kaveh Amouzgar, Mirza Cenanovic, Kent Salomonsson
Multi-objective optimization of material model parameters of an adhesive layer by using SPEA2

Cenanovic and Amouzgar took part in planning, implementing the procedure and writing the paper. Salomonsson provided the original data and advice during the work.

Acknowledgements

The work presented in this thesis was carried out during the years 2012-2015 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 Professor Kent Salomonsson for his support 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 my colleagues at the department for the easygoing atmosphere that each and everyone created. Lastly, I would like to thank my family and friends.

Contents

Introduction

Introduction and motivation

Surface problems are found in many places in industry. These include, for instance, sheet metal forming which can be modelled as shells, membranes such as sails and structures, heat transfer accross a curved geometry, distance between two points on a curved surface (also called geodesic distance field [8]), for form finding, curvature driven problems and composite materials where thin layers add stiffness to the bulk (sandwich constructions in airplane industry and glue laminated timber). In order to create models of these types of surface problems it is important to have an analytical model that can be used to verify the numerical model. Another important property of surface models is to be general and rapidly implementable.

Traditionally three dimensional surface problems were modeled in a two dimensional parameter space and mapped to the real surface in three dimensions [7, 6], also known as the parametrization of a surface or parametric approach to surface problems. This requires the use of an exact surface representation and additionally differential operators that are defined on a curved space using base vectors. 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 parametrized 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 [24].

An approach of modeling finite element methods for surface partial differential equations (PDEs) without the use of parametric mapping was used by Dziuk [13] for the Beltrami operator, 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 operator is used by Delfour and Zolésio in [10, 11, 12], 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 [16], where a FEM was created for a general curved linear membrane shell, and Hansbo, Larson and Larsson in [18] for large deformations theory. A recent overview of FEM for surface PDEs is given by Dziuk and Elliot in [15], where the tangential approach is applied on a large set of surface PDEs and discussed in the paper and references therein. The ideas by Dziuk and Elliot are applied on flat triangular elements and there is a need to develop more general finite element methods.

For evolving surface problems a novel FEM was proposed by Olshanskii, Reusken and Grande [20] 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 basis 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 Chapter 3 of this thesis for a detailed overview of some recent work on surface and fictitious domain problems, called the *cut finite element method* (CutFEM). 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. [3, 4, 17]. The work done in this thesis further develops surface PDEs using the tangential calculus approach within the CutFEM framework, see Supplements 1 and 2.

Aim and limitations

The purpose of this thesis is to give a brief overview of the ideas of the various approaches used in the supplements and to show the details of the implementations. The implementations and their details are limited to small proof of concept scripts in the high level language MATLAB[1] and are not intended for production code without rework. The algorithms provided in the Supplements 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 introduced herein.

[1]http://mathworks.com/products/matlab/

1 Verification and Validation

CHAPTER INTRODUCTION

Solving partial differential equations on surfaces is done by creating computational models that are based on the finite element method and approximate the true solution field. How good this approximation is depends on the errors that are introduced. In order to make computational methods comparable we need a way of estimating the error. This chapter gives an introduction to the different types of errors that are introduced in various ways and gives an overview of ways to verify and validate the approximations that introduce these errors.

The analysis of the error consists of two aspects:

1. Verification:
   Checking whether the model returns expected results.
2. Validation:
   Checking how well the model describes reality.

1.1 Modeling errors

Modeling errors are introduced by idealization of the "true" physical system to a simplified system - a mathematical system. Computational models are only as good as the mathematical models that they are based on. Choosing the right mathematical model for a certain application is not trivial due to a trade off which the user must make. A larger, more specific and thorough model will be closer to reality, but the mathematical system will most certainly be more complicated and take longer time to solve than a simpler and less accurate model. Depending on how accurately the user wants to model a particular application there might exist several different models to choose from, depending on what is asked of the model and how accurate the answer is supposed to be.

1.2 Discretization errors

A mathematical model is, though a simplification of reality, still hard to solve. The models contain partial differential equations, often coupled, and can be linear or non-linear. Finite element methods introduce a discretization error that can be analyzed and used as a tool for choosing between different FE-methods.

1.3 Supermodels

Partial differential equations on surfaces can be modeled on a three dimensional manifold. The corresponding three-dimensional mathematical model is called a *supermodel* which is used to verify other simpler surface models by trying to prove that the models are equivalent under certain conditions. The proof can be mathematical (if possible) or numerical in the sense that a solution field is compared from both models and the error is analyzed.

Similar to a supermodel, a *representative volume element* (RVE) can be created and experimentally validated in order to be used as verification for other simpler, homogeneous models. The validation of an RVE is done experimentally and usually involves tweaking of RVE specific parameters, see Supplement 3.

1.4 Method of Manufactured Solutions

A method used to verify the computational model is the so called method of manufactured solutions. It is used everywhere but yet is seldom explained in the literature, maybe due to its seemingly trivial nature. The method involves assuming a solution field and then working out the analytical right hand side of a linear equation, initial conditions and boundary conditions that fit that solution field. The right hand side, initial- and boundary conditions are then used in the computational algorithm to arrive at a solution. The error between the computed solution and the assumed solution is evaluated to determine how well the computational model is compared to other models. An example can be seen in Section 4.5.

1.5 Validation

Since not all problems are linear, using the method of manufactured solutions is not always possible to verify the computational model. In order to test the validity of such a model, physical tests must be carried out and then compared to the simulated results from the computational model. If the observations agree with the predictions, the model is considered accurate. See Supplement 2 for techniques regarding validation.

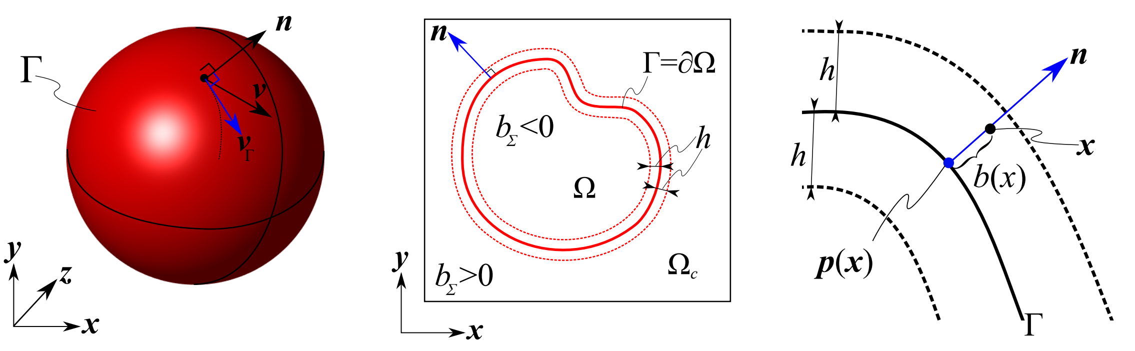
2 Surface problems

CHAPTER INTRODUCTION

This chapter introduces surface problems that are modeled using the *tangential approach* which is one of the central aspects to this thesis. An overview of this approach is given. The two main surface representations are introduced.

2.1 Tangential calculus

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 [13] and analyzed for shells by Delfour and Zolesio in [10, 11, 12] and adapted by Hansbo and Larson in [16] for a general membrane shell model. Traditionally, surface problems were modeled in a parametric setting and mapped to Cartesian three dimensional space using various differential operators, see [7, 6]. 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 [11].



(a) 2D surface in 3D (b) Surface manifold (c) Tubular neighborhood Ω_h

Figure 2.1: Surface Manifold

Consider a smooth $d-1$-dimensional surface Γ embedded in $\mathbb{R}^d$, where d is typically 2 or 3, and contained by Ω which is a subset of $\mathbb{R}^d$. The surface is modeled as a thin domain around Γ with the thickness $2h$. It is assumed that the thickness is small relative to the size of the surface. A signed distance function $b : \mathbb{R}^d \to \mathbb{R}$ is given by,

$$b(\boldsymbol{x}) = \begin{cases} b(\boldsymbol{x}) < 0 & \text{in } \Omega \\ b(\boldsymbol{x}) = 0 & \text{on } \Gamma \\ b(\boldsymbol{x}) > 0 & \text{in } \Omega_c \end{cases} \quad \forall \boldsymbol{x} \in (\Omega \cup \Omega_c), \tag{2.1}$$

where $\Omega_c = \{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{x} \notin \Omega\}$. If the surface is sufficiently smooth, then for a small $h > 0$ a neighborhood around Γ can be defined as

$$\Omega_h = \{\boldsymbol{x} \in \mathbb{R}^d : |b(\boldsymbol{x})| < h\}, \tag{2.2}$$

where $|\nabla b| = 1$ and $b(\boldsymbol{x}) = \boldsymbol{n}(\boldsymbol{x})$ for $\boldsymbol{x} \in \Gamma$.

The nearest point projection map $\boldsymbol{p} : \Omega_h \to \Gamma$ is given by

$$\boldsymbol{p}(\boldsymbol{x}) = \boldsymbol{x} - b(\boldsymbol{x})\boldsymbol{\nabla} b(\boldsymbol{x}) \tag{2.3}$$

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

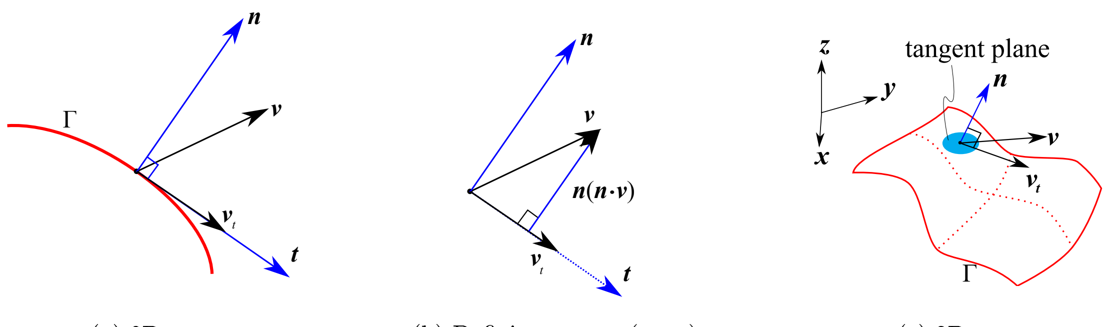
$$\frac{\partial}{\partial x_1} p_1 = 1 - \frac{\partial}{\partial x_1}\left(b(\boldsymbol{x})\frac{\partial b(\boldsymbol{x})}{\partial x_1}\right) = 1 - \frac{\partial b(\boldsymbol{x})}{\partial x_1}\frac{\partial b(\boldsymbol{x})}{\partial x_1} - b(\boldsymbol{x})\frac{\partial^2 b(\boldsymbol{x})}{\partial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

where $b(\boldsymbol{x}) = 0$ for $\boldsymbol{x} \in \Gamma$ so $b(\boldsymbol{x})\dfrac{\partial^2 b(\boldsymbol{x})}{\partial x_1^2} = 0$ on the surface. Thus for a $\boldsymbol{x} \in \Omega_h$, the *linear projector onto the tangent plane* at $\boldsymbol{p}(\boldsymbol{x})$ is given by

$$D\boldsymbol{p}(\boldsymbol{x}) = \boldsymbol{I} - \boldsymbol{\nabla} b(\boldsymbol{x}) \otimes \boldsymbol{\nabla} b(\boldsymbol{x}) =: \boldsymbol{P}_\Gamma(\boldsymbol{x}), \tag{2.4}$$

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

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 notational convenience the subscript of $\boldsymbol{P}_\Gamma$ will frequently be omitted. Note that the operator is natural in tensor analysis, see e.g. [19], where it is defined as the projection of a function $\boldsymbol{v}$ onto the plane defined by the normal $\boldsymbol{n}$ see Figure 2.2.



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

Figure 2.2: Projection of $\boldsymbol{v}$ onto the surface Γ

Using the definition in Figure 2.2b, we have

$$\boldsymbol{v}_t = \boldsymbol{v} - \boldsymbol{v}_n = \boldsymbol{v} - \boldsymbol{n}(\boldsymbol{n} \cdot \boldsymbol{v}). \tag{2.5}$$

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

$$(\boldsymbol{a} \otimes \boldsymbol{b})\boldsymbol{c} = \boldsymbol{a}(\boldsymbol{b} \cdot \boldsymbol{c}), \tag{2.6}$$

(which is what Dziuk used in [13]) and get

$$\boldsymbol{v}_t = \boldsymbol{v} - (\boldsymbol{n} \otimes \boldsymbol{n})\boldsymbol{v}, \tag{2.7}$$

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

$$\boldsymbol{v}_t = (\boldsymbol{I} - \boldsymbol{n} \otimes \boldsymbol{n})\boldsymbol{v} = \boldsymbol{P}\boldsymbol{v}, \tag{2.8}$$

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 1. 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,

$$\boldsymbol{v}_t = \boldsymbol{P}\boldsymbol{v} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \begin{pmatrix} 0 \\ v_y \\ v_z \end{pmatrix}. \tag{2.9}$$

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.

Remark 2. 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

$$\boldsymbol{V} = \begin{pmatrix} V_{xx} & V_{xy} & V_{xz} \\ V_{yx} & V_{yy} & V_{yz} \\ V_{zx} & V_{zy} & V_{zz} \end{pmatrix}, \qquad \boldsymbol{P} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{2.10}$$

then

$$\boldsymbol{V}_t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_{xx} & V_{xy} & V_{xz} \\ V_{yx} & V_{yy} & V_{yz} \\ V_{zx} & V_{zy} & V_{zz} \end{pmatrix} = \begin{pmatrix} V_{xx} & V_{xy} & V_{xz} \\ 0 & 0 & 0 \\ V_{zx} & V_{zy} & V_{zz} \end{pmatrix}. \tag{2.11}$$

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.

$$\boldsymbol{V}_t^P = \boldsymbol{P}\boldsymbol{V}\boldsymbol{P} = \begin{pmatrix} V_{xx} & V_{xy} & V_{xz} \\ 0 & 0 & 0 \\ V_{zx} & V_{zy} & V_{zz} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} V_{xx} & 0 & V_{xz} \\ 0 & 0 & 0 \\ V_{zx} & 0 & V_{zz} \end{pmatrix}, \tag{2.12}$$

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.

2.1.1 The discrete tangential projection

In the discrete setting where the surface is discretized in a piecewise linear fashion 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 2.3. This means that we have have introduced another discretization error through the projection operator. For a in depth discussion on this error see [4].

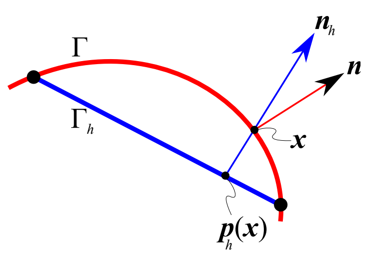


Figure 2.3: Discrete normal $\boldsymbol{n}_h$ compared to exact surface normal $\boldsymbol{n}$

2.2 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. 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 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.

Implicit surfaces on the other hand can be used to represent complex evolving surfaces without the need of re-meshing. 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., [1, 5]. Another way is to use analytical implicit surfaces descriptions, see Burman et al. [3] for an overview. Complex geometries can be created from simple geometries using analytical functions together with boolean operations. Simple boundaries can be created by embedding the surfaces into the domains such that the boundaries are defined by the borders of the domain.

An implicit surface ϕ is visualized using the discrete faces that are generated by intersecting ϕ with a mesh, see e.g., Figure 2.4.

The following analytical surface descriptions are used in the supplements.

Cylinder function:

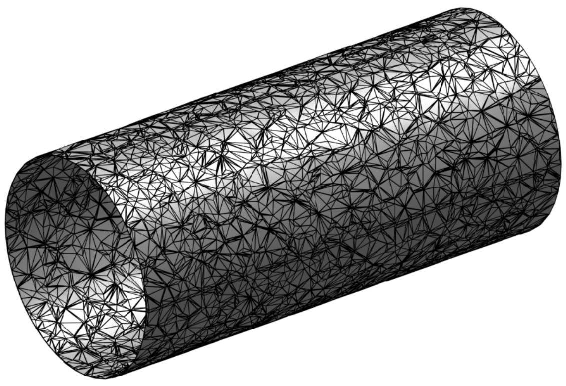
$$\phi(x, y, z) = \sqrt{(x - x_c)^2 + (y - y_c)^2} - r$$

where $[x_c, y_c]$ is the center of the cylinder. See Figure 2.4a.

Oblate spheroid:

$$\phi(x, y, z) = x^2 + y^2 + (2z)^2 - 1$$

See Figure 2.4b.


(a) Cylinder

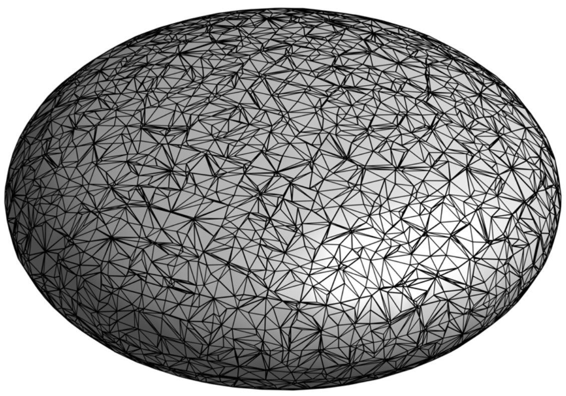

(b) Oblate spheroid

Figure 2.4: Implicit surfaces

3 Cut Finite Element Method

CHAPTER INTRODUCTION

The cut finite element method (*CutFEM*) 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 [20] where a new technique was introduced for hyper-surfaces.

CutFEM is a general method for dealing with complicated cases involving interface problems, complex and evolving geometries, see Burman et al. [3] for a review of CutFEM, where it is applied on a broader range of problems, not just surface problems. For surface problems the idea of CutFEM is to use a higher dimensional background mesh to discretize the surface problem but integrate only over the intersection between the background mesh and the discrete surface. This approach is suitable for evolving surface problems such as the one in Supplement 1, but it also allows for embedding arbitrarily shaped reinforcement to an elastic domain. The CutFEM approach needs stabilization due to ill-conditioning of the linear system (see [4] for details) and model dependent instability (see, e.g., [17]). The following sections will introduce the idea in the finite element space and provide an overview of the stabilization.

3.1 Domain

Let $\boldsymbol{x}$ denote Euclidean co-ordinates in $\mathbb{R}^d$, where $d = 2$ or 3, such that $\boldsymbol{x} \in \Omega$, where Ω is a bounded domain. Let Γ denote the smooth $d-1$-dimensional interface contained by Ω and dividing the domain 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 3.1.

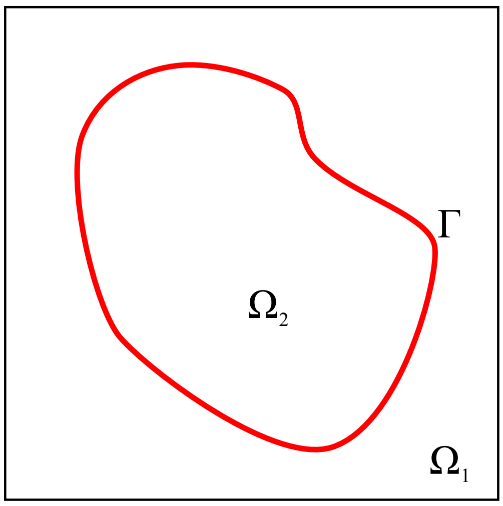


Figure 3.1: 2D representation of the problem domain

3.2 Discretization

The discretization is done by letting $\tilde{\mathcal{K}}_h$ be a tessellation using polyhedrons (3D) or polygons (2D) of the domain Ω, with an element size parameter $0 < h < h_0$, containing Γ. Note that Γ needs to be sufficiently smooth in order not to intersect the same element more than once, i.e., the mesh size parameter must be chosen small enough to resolve the curvature of Γ. Recalling the interface representation in Section 2.1, we let ϕ_h denote the continuous piecewise linear approximation of the signed distance function ϕ to define the discrete surface Γ_h as the zero level set of ϕ_h such that

$$\Gamma_h = \{\boldsymbol{x} \in \Omega : \phi_h(\boldsymbol{x}) = 0\} \tag{3.1}$$

and note that Γ_h is a flat polygon (3D case) or a line segment (2D case). Let $\boldsymbol{n}_h$ denote the piecewise constant outward unit normal to Γ_h. The Dirichlet boundary to the interface is denoted by $\partial\Gamma_{h,D}$. In the case of surface problems, 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

$$\mathcal{K}_h = \{K \in \tilde{\mathcal{K}}_h : K \cap \Gamma = \emptyset\}, \tag{3.2}$$

see Figure 3.2.

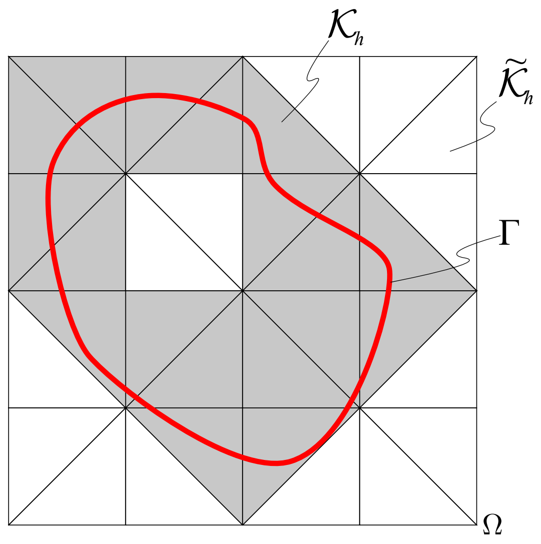


Figure 3.2: Interface elements in 2D

3.3 Dirichlet conditions

A simple way of dealing with Dirichlet boundary conditions in this setting is to directly prescribe the degrees of freedom on the background mesh. If $\partial\Gamma_{h,D}$ is intersecting the boundary of Ω then we can denote this by $\partial\mathcal{K}_{h,D}$, see Figure 3.3. This situation occurred in Supplement 1 and was straightforward. If however $\partial\Gamma_{h,D}$ is completely contained by Ω then the conditions are mesh-dependent and $\partial\mathcal{K}_{h,D}$ must be carefully constructed or the mesh modified to facilitate proper Dirichlet conditions. This was the situation in Supplement 2 and was straightforward for the pulled cylinder problem, but in the case of the oblate spheroid problem, the mesh needed modification since it was unstructured. A more sophisticated method is needed when dealing with more complex situations.

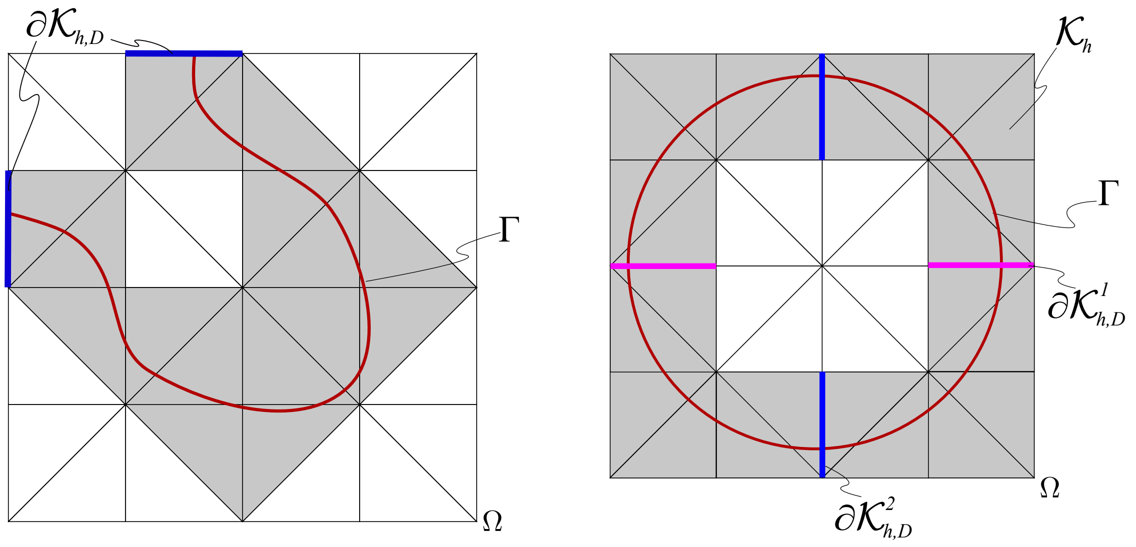
3.4 Finite element space

In the setting of the CutFEM 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.

Let the finite element space be defined as the space of continuous piecewise linear (triangles in 2D and tetrahedra in 3D), bi-linear (quadrilaterals in 2D) or tri-linear (hexahedra in 3D) denoted as $\tilde{V}_h$ and defined on $\tilde{\mathcal{K}}_h$. The corresponding discrete space V_h is defined on $\mathcal{K}_h$ by

$$V_h = \{\boldsymbol{v} \in \tilde{V}_h|_{\Omega_h} : \boldsymbol{v} = \mathbf{0} \text{ on } \partial\tilde{\mathcal{K}}_{h,D}\}. \tag{3.3}$$

Consider for example diffusion on a surface


$\partial\mathcal{K}_{h,D}$
Γ
Ω
$\mathcal{K}_h$
Γ
$\partial\mathcal{K}^1_{h,D}$
Ω
$\partial\mathcal{K}^2_{h,D}$

(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 chosen carefully

Figure 3.3: Handling Dirichlet boundaries

$$-\nabla_\Gamma \cdot \nabla_\Gamma \boldsymbol{u} = \boldsymbol{f} \quad \text{on } \Gamma, \tag{3.4}$$

where

$$\nabla_\Gamma \boldsymbol{u} = \boldsymbol{P}\nabla\boldsymbol{u}, \tag{3.5}$$

and

$$\nabla_\Gamma \cdot \boldsymbol{v} = \mathrm{tr}(\nabla_\Gamma \otimes \boldsymbol{v}) \tag{3.6}$$

The finite element method on Γ_h is then given by: find the solution field $\boldsymbol{u}_h \in V_h$ such that

$$a_h(\boldsymbol{u}_h, \boldsymbol{v})_{\Gamma_h} = l_h(\boldsymbol{v})_{\Gamma_h} \quad \forall \boldsymbol{v} \in V_h, \tag{3.7}$$

where

$$a_h(\boldsymbol{u}_h, \boldsymbol{v})_{\Gamma_h} = \int_{\Gamma_h} \nabla_{\Gamma_h}\boldsymbol{u} \cdot \nabla_{\Gamma_h}\boldsymbol{v} ds \tag{3.8}$$

and

$$l_h(\boldsymbol{v})_{\Gamma_h} = \int_{\Gamma_h} \boldsymbol{f}\boldsymbol{v} ds \tag{3.9}$$

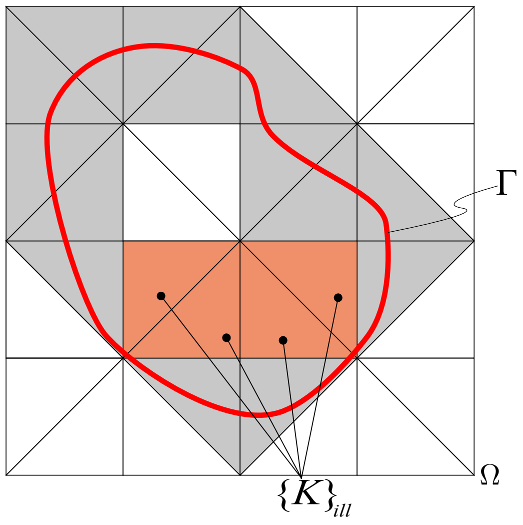


Figure 3.4: Elements $\{K\}_{ill}$ that cause ill-conditioning of the linear system.

3.5 Stabilization

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 3.4. This leads to a severely ill-conditioned linear system of (3.7) that needs to be addressed see e.g. [4]. Furthermore a finite element method of surface problems using higher dimensional shape functions can be unstable (see, e.g., [17], Supplement 1 and 2). For these reasons a stabilization method is introduced. We begin by introducing additional quantities on the mesh.

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 K^N share a face, see Figure 3.5. Define a set of interior faces of $\mathcal{K}_h$ by

$$\mathcal{F}_h = \{F = K \cap K^N : K, K^N \in \mathcal{K}_h\}. \tag{3.10}$$

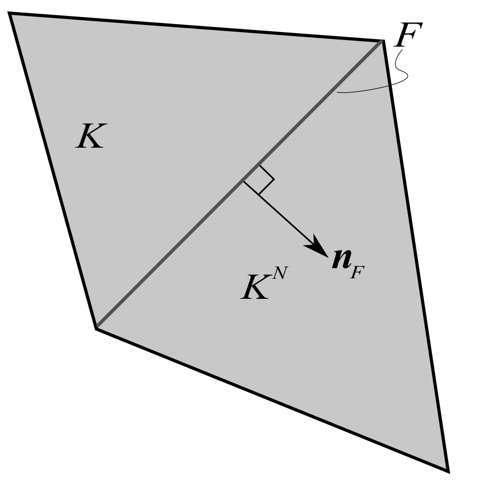
Each internal face defines a unit normal vector $\boldsymbol{n}_F$ that is perpendicular to the face and oriented exterior to K , see Figure 3.5. The jump of the gradient of $\boldsymbol{v}$ across the face F of elements K and K^N is defined on each side of F as

$$[\![\partial_{nF}\boldsymbol{v}]\!] = \boldsymbol{n}_F \cdot \nabla\boldsymbol{v}|_{F\in K} - \boldsymbol{n}_F \cdot \nabla\boldsymbol{v}|_{F\in K^N}, \tag{3.11}$$

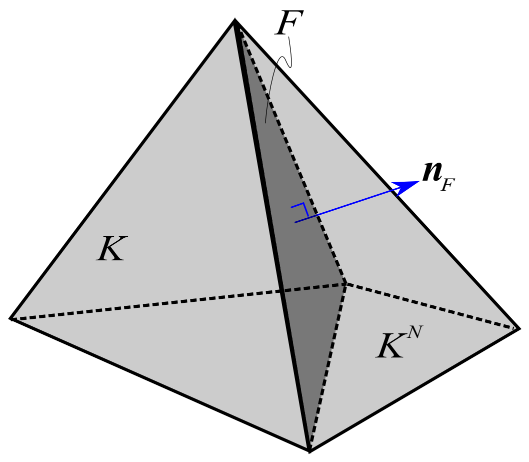
where $\nabla\boldsymbol{v}|_{F\in K}$ denotes the gradient on the face F of element K. The stabilizing term $j_h(\cdot,\cdot)$ is then given as the sum of all gradient jumps as

$$j_h(\boldsymbol{v},\boldsymbol{w}) = \sum_{F\in\mathcal{F}_h} \int_F \gamma[\![\partial_{nF}\boldsymbol{v}]\!] \cdot [\![\partial_{nF}\boldsymbol{w}]\!] ds, \tag{3.12}$$

where γ is a positive constant that penalizes the jump. The stabilized form of (3.7) becomes



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



(b) 3D case

Figure 3.5: Interior faces of $\mathcal{K}_h$

$$A_h(\boldsymbol{u}_h, \boldsymbol{v}) = l_h(\boldsymbol{v}) \quad \forall \boldsymbol{v} \in V_h, \tag{3.13}$$

where the bilinear form $A_h(\cdot, \cdot)$ is defined by

$$A_h(\boldsymbol{v}, \boldsymbol{w}) = a_h(\boldsymbol{v}, \boldsymbol{w})_{\Gamma_h} + j_h(\boldsymbol{v}, \boldsymbol{w}) \quad \forall \boldsymbol{v}, \boldsymbol{w} \in V_h. \tag{3.14}$$

4 Implementation Details

4.1 Choice of software

The 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 analysis and optimization of the implementation. There is thus much room for optimizing the performance of the algorithm and implementation. 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[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 in the assembly process, where an indexed approach [9] is utilized.

4.2 Zero level surface approximation

Starting from a mesh $\tilde{\mathcal{K}}$ in $\mathbb{R}^d$ on the domain Ω in which the implicit surface Γ is embedded we have the approximate piecewise linear signed distance function $\phi(\boldsymbol{x})$ (in the following we make no distinction between ϕ and ϕ_h) such that $\Gamma_h = \{\boldsymbol{x} \in \Omega : \phi(\boldsymbol{x}) = 0\}$, see Section 2.2 for details on how to construct implicit surfaces. The overall procedure is to use linear interpolation on the discrete ϕ values for each element edge in order to find the zero level surface point, see Figure 4.3. What follows is an algorithm describing how to extract the discrete surface Γ_h and the corresponding set of normals $\{\boldsymbol{n}_h\}$.

1. Find the set of elements $\mathcal{K}_h$, referred to as the background mesh, that is intersected by the

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

surface by using the discrete distance values ($\phi > 0$ and $\phi < 0$) in some nodes of element K_i, see Figures 4.1 and 4.2.

2. 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 $\phi|_{e_i}$ to determine if the edge is cut.

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

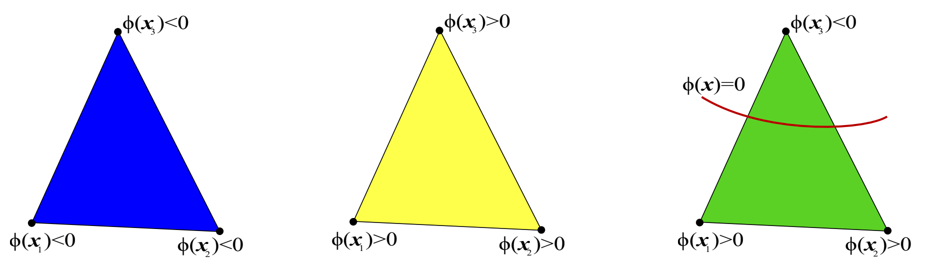
   c) Let $\mathbf{x}_{e_i,1} = \mathbf{x}_{e_i}|_{\phi|_{e_i}>0}$ (the coordinate corresponding to the highest value of ϕ) and $\mathbf{x}_{e_i,0} = \mathbf{x}_{e_i}|_{\phi|_{e_i}<0}$ and compute the vector $\mathbf{n}_i = \mathbf{x}_{e_i,1} - \mathbf{x}_{e_i,0}$. See figure 4.3b.

3. Compute the element vector $\boldsymbol{n}_\phi = \sum \boldsymbol{n}_i$. Note that $\boldsymbol{n}_\phi$ is pointing in the general direction of $\nabla\phi$ and is only used to determine the orientation of the face normals.

4. Depending on the number of nodes in element K_i^Γ and the orientation of the cut, several cut cases must be considered, see figure 4.4 for tetrahedral element and figure 4.5 for hexahedra.

5. 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 4.6.

6. 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 4.4.2 for a discussion on the topology) and for visualization (easier handling and interpolated surface shading).


$\phi(\boldsymbol{x}_3)<0$
$\phi(\boldsymbol{x}_1)<0$
$\phi(\boldsymbol{x}_2)<0$
$\phi(\boldsymbol{x}_3)>0$
$\phi(\boldsymbol{x}_1)>0$
$\phi(\boldsymbol{x}_2)>0$
$\phi(\boldsymbol{x}_3)<0$
$\phi(\boldsymbol{x})=0$
$\phi(\boldsymbol{x}_1)>0$
$\phi(\boldsymbol{x}_2)>0$

(a) All ϕ values are inside the interface.

(b) All ϕ values are outside the interface.

(c) ϕ is both negative and positive inside an element.

Figure 4.1: Element categories

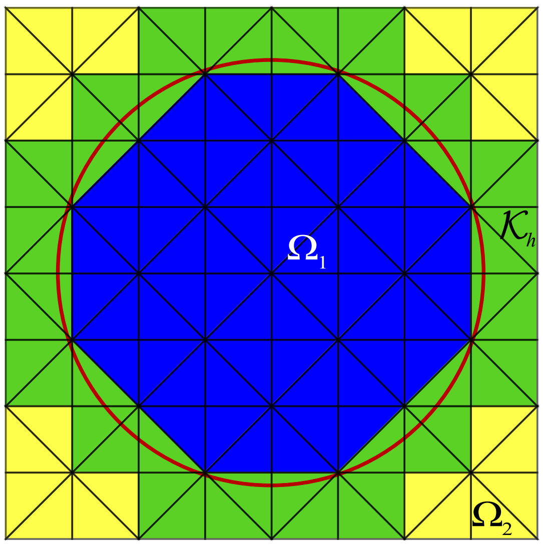
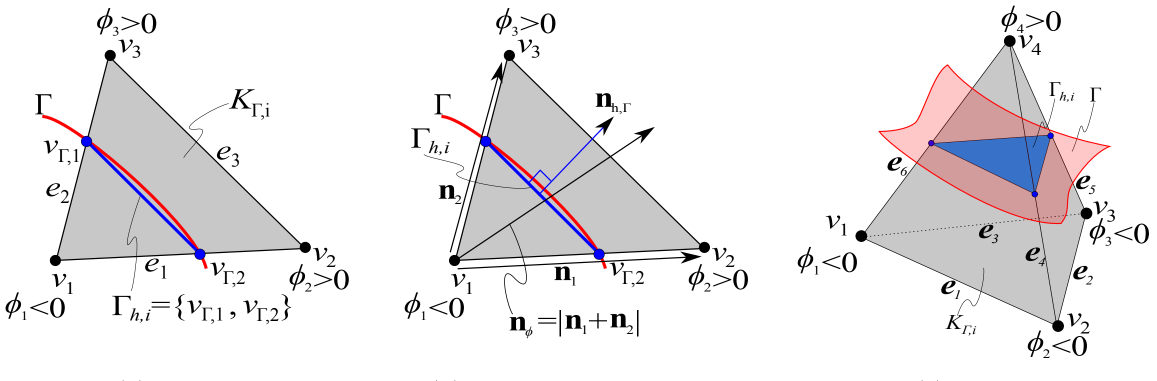
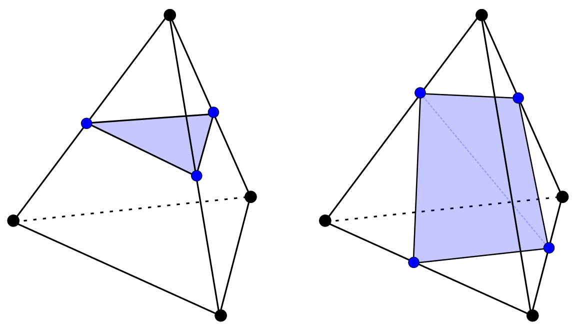


Figure 4.2: Mesh categorisation



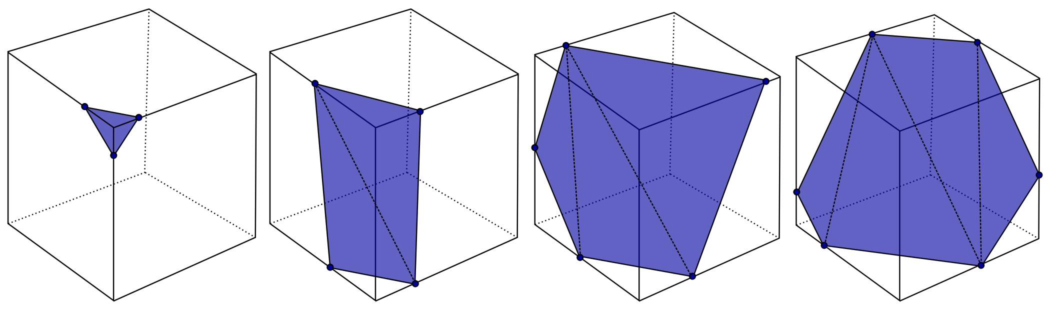
(a) 2D case (b) Surface element normal (c) 3D case

Figure 4.3: Surface element Γ_h^i and parent element K_Γ^i in 2D and 3D



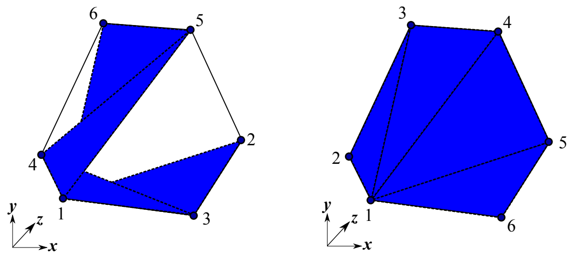
(a) Triangular (b) Quadliteral

Figure 4.4: Tetrahedral cut cases



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

Figure 4.5: Hexahedra cut cases



(a) Wrong numbering (b) Proper numbering order

Figure 4.6: Numbering order for arbitrary polygon in $\mathbb{R}^3$. The order without rotation to $\mathbb{R}^2$ is shown in a). In sub-figure 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.

4.3 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 Supplement 1 to compute the curvature of a surface and iteratively find the design that minimizes it. In the following we drop the distinction between Γ and Γ_h. Recall, however, that the FEM is discretized replacing Γ with Γ_h.

In Supplement 1 the minimal surface problem is modeled as a level set surface that is incrementally advected by a velocity field,

$$\dot{\boldsymbol{x}}_\Gamma = \Delta_\Gamma \boldsymbol{x}_\Gamma = -2H\boldsymbol{n}. \tag{4.1}$$

Here Δ_Γ denotes the Laplace-Beltrami operator defined by

$$\Delta_\Gamma = \nabla_\Gamma \cdot \nabla_\Gamma, \tag{4.2}$$

where ∇_Γ is the surface gradient given by

$$\nabla_\Gamma = \boldsymbol{P}\nabla. \tag{4.3}$$

The curvature measure is given by

$$H = \frac{\kappa_1 + \kappa_2}{2}, \tag{4.4}$$

see e.g. [2].

The surface is deformed by advection, see e.g., [22, 21] for more information on the level set method. The level set function ϕ is updated using the computed velocity from (4.1) by

$$\frac{d\phi}{dt} = \frac{\partial \phi}{\partial t} + \dot{\boldsymbol{x}}_\Gamma \cdot \nabla\phi = 0 \tag{4.5}$$

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. This is made possible by driving the surface using advection (4.5). The discretization of surface problems using implicit surface representations can be done using the cut finite element method, see Section 3.

An interesting aspect of this is to use adaptivity to refine the mesh in areas close to the boundary or where the curvature is high.

An interesting area for future work is form finding coupled with topology optimization.

For the normal curvature flow from (4.1) we have

$$(\dot{\boldsymbol{x}}_\Gamma, \boldsymbol{v})_\Gamma + j(\dot{\boldsymbol{x}}_\Gamma, \boldsymbol{v}) = a\,(\boldsymbol{x}_\Gamma, \boldsymbol{v})_\Gamma \quad \forall \boldsymbol{v} \in V_h, \tag{4.6}$$

where

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

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

$$\int_\Gamma \frac{d\boldsymbol{x}}{dt} \cdot \boldsymbol{v} d\Gamma + \sum_{F \in \mathcal{F}} \int_F [\boldsymbol{n}_F \cdot \nabla \dot{\boldsymbol{x}}_\Gamma] \cdot [\boldsymbol{n}_F \cdot \nabla \boldsymbol{v}] ds = -\int_\Gamma \nabla \boldsymbol{x}_\Gamma \cdot \nabla \boldsymbol{v} d\Gamma \tag{4.7}$$

which leads to the discrete system

$$\mathbf{M}\frac{1}{k_n}(\mathbf{x}_{n+1} - \mathbf{x}_n) = -\mathbf{S}\mathbf{x}_{n+1}, \tag{4.8}$$

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

$$\mathbf{M} = \int_\Gamma \mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi} d\Gamma + \sum_{F \in \mathcal{F}} \int_F [\boldsymbol{n}_F \cdot \nabla \dot{\boldsymbol{x}}_\Gamma] \cdot [\boldsymbol{n}_F \cdot \nabla \boldsymbol{v}] ds, \tag{4.9}$$

and

$$\mathbf{\Phi} = \begin{bmatrix} \varphi^1 & 0 & 0 & \varphi^2 & 0 & 0 & \dots \\ 0 & \varphi^1 & 0 & 0 & \varphi^2 & 0 & \dots \\ 0 & 0 & \varphi^1 & 0 & 0 & \varphi^2 & \dots \end{bmatrix}, \tag{4.10}$$

with $\boldsymbol{\varphi}$ denoting the basis functions of V_h, $\mathbf{x}_n$ denotes the nodal surface values, i.e., the coordinates of the underlying mesh, at the current virtual time step n of the narrow band of the mesh around the interface and $\mathbf{x}_{n+1}$ denotes the nodal values at the next virtual time step, k_n is the time step length and $\mathbf{S}$ is the stiffness matrix given on Voigt form by

$$\mathbf{S} = \int_\Gamma \mathbf{B}_\Gamma^{\mathrm{T}} \mathbf{B}_\Gamma d\Gamma, \tag{4.11}$$

where

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

and $\varphi^1_{\Gamma,x}$ denotes the tangential x derivative of the first basis function since we have

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

with

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

and $\varphi^1_x = \dfrac{\partial \varphi_1}{\partial x}$. The discrete Laplace equation (4.8) can be rewritten according to

$$\mathbf{M} \frac{1}{k_n} \mathbf{x}_{n+1} - \mathbf{M} \frac{1}{k_n} \mathbf{x}_n = -\mathbf{S} \mathbf{x}_{n+1} \tag{4.12}$$

$$\Leftrightarrow \mathbf{M} \frac{1}{k_n} \mathbf{x}_n = \mathbf{S} \mathbf{x}_{n+1} + \mathbf{M} \frac{1}{k_n} \mathbf{x}_{n+1}, \tag{4.13}$$

where (4.13) was proposed by Dziuk in [14]. We need to compute $\mathbf{x}_{n+1}$ by

$$\mathbf{M}\frac{1}{k_n}\mathbf{x}_n = \left(\mathbf{S} + \mathbf{M}\frac{1}{k_n}\right)\mathbf{x}_{n+1} \Leftrightarrow \tag{4.14}$$

$$\mathbf{x}_{n+1,} = \left(\mathbf{S} + \mathbf{M}\frac{1}{k_n}\right)^{-1}\mathbf{M}\frac{1}{k_n}\mathbf{x}_n. \tag{4.15}$$

The surface velocity field towards the lower curvature (around the interface Γ) is then given by

$$\mathbf{v}_n = \frac{\mathbf{x}_{n+1} - \mathbf{x}_n}{k_n}. \tag{4.16}$$

4.3.1 Level set advection

The level set advection equation is given by

$$\frac{\partial \phi}{\partial t} + \dot{\boldsymbol{x}}_\Gamma \cdot \nabla \phi = 0. \tag{4.17}$$

The time derivative of the distance function is discretized by

$$\frac{\partial \phi}{\partial t} \approx \frac{\phi_{n+1} - \phi_n}{k_n},$$

which yields the updated implicit surface

$$\phi_{n+1} = \phi_n + k_n \mathbf{v}_n \cdot \nabla \phi. \tag{4.18}$$

A finite element method is used in order to compute the discontinuous gradient field $\nabla\phi_h$ on the narrow band $\mathcal{K}_{h,N}$, see Figure (4.7) .

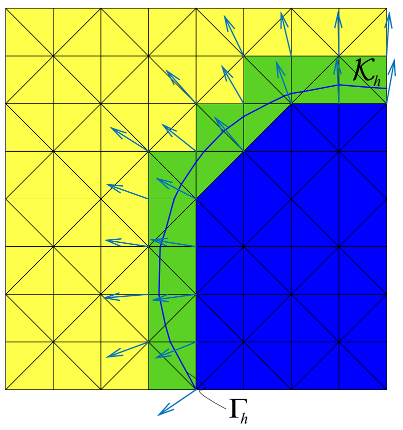


Figure 4.7: Discontinuous gradient field on $\mathcal{K}_h$

We begin by computing the element gradient of the distance function $\nabla\phi_e$ and then projecting it to the nodes on the grid using an L_2 projection (see Section (4.3.2) for details on the L_2 projection).

$$\nabla\phi = \mathbf{M}^{-1}\int_{\mathcal{K}_{h,N}} \nabla\phi_e \cdot \boldsymbol{v} dV$$

where $\nabla\phi_e$ is the element gradient of ϕ given by

$$\nabla\phi_e = \left(\int_K \phi_K \cdot \nabla\varphi_K dK\right).$$

Finally $\nabla\phi$ has to be normalized by

$$\nabla\phi = \frac{\nabla\phi}{|\nabla\phi|}.$$

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 (4.18) the property $|\nabla\phi| = 1$ gets degraded and the distance function needs to be reinitialized. This can be done in various ways e.g. [23, 22] the naive approach is to compute the closest distance from each node on the narrow band of the mesh to the discrete interface.

4.3.2 The L_2 projection

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

$$(\boldsymbol{u}_h, \boldsymbol{v}) = (\boldsymbol{u}, \boldsymbol{v}) \quad \forall \boldsymbol{v} \in V_h.$$

See Figure (4.8). Or equivalently

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

where

$$\mathbf{f} = \int_\Omega \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{U} d\Omega,$$

with $\mathbf{U}$ being the element value for $\boldsymbol{u}$ and letting $\boldsymbol{\Phi}$ be the Voigt form of the basis functions in V_h similar to (4.10) we have

$$\mathbf{M} = \int_\Omega \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} d\Omega.$$

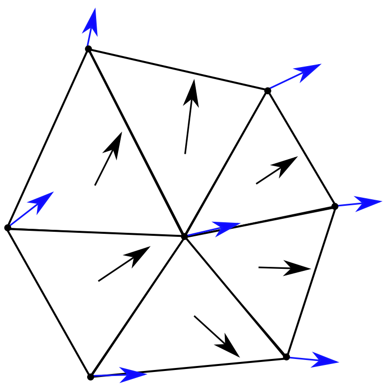


Figure 4.8: L_2 projection for averaging the element values (black arrows) onto the nodal values (blue arrows)

4.4 Linear Elastic Membrane Shell

4.4.1 Surface strain and stress

The surface strain tensor is defined by

$$\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u}) := \frac{1}{2}\left(\nabla_\Gamma \otimes \boldsymbol{u} + (\nabla_\Gamma \otimes \boldsymbol{u})^{\mathrm{T}}\right) \tag{4.19}$$

and the in-plane strain tensor is given by:

$$\boldsymbol{\varepsilon}_\Gamma^P(\boldsymbol{u}) := \boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u}) - ((\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u}) \cdot \boldsymbol{n}) \otimes \boldsymbol{n} + \boldsymbol{n} \otimes (\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u}))) \tag{4.20}$$

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

$$-\nabla_\Gamma \cdot \boldsymbol{\sigma}_\Gamma^P(\boldsymbol{u}) = \boldsymbol{f} \quad \text{on} \quad \Gamma \tag{4.21}$$

$$\boldsymbol{\sigma}_\Gamma^P(\boldsymbol{u}) = 2\mu\boldsymbol{\varepsilon}_\Gamma^P + \lambda_0 \mathrm{tr}\boldsymbol{\varepsilon}_\Gamma^P \boldsymbol{P}_\Gamma \quad \text{on} \quad \Gamma \tag{4.22}$$

$$\boldsymbol{u} = \boldsymbol{0} \quad \text{on} \quad \partial\Gamma_D \tag{4.23}$$

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

$$\lambda_0 := \frac{2\lambda\mu}{\lambda + 2\mu} = \frac{E\nu}{1-\nu^2}, \; \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \; \mu = \frac{E}{2(1+\nu)} \tag{4.24}$$

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

The bilinear form of 4.21 is

$$a_\Gamma(\boldsymbol{u},\boldsymbol{v}) = l_\Gamma(\boldsymbol{v}) \quad \forall \boldsymbol{v} \in V_h, \tag{4.25}$$

where

$$a_\Gamma(\boldsymbol{u},\boldsymbol{v}) = (2\mu\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u}),\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{v}))_\Gamma - (2\mu\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u})\cdot\boldsymbol{n},\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{v})\cdot\boldsymbol{n})_\Gamma + (\lambda_0\nabla_\Gamma\cdot\boldsymbol{u},\nabla_\Gamma\cdot\boldsymbol{v})_\Gamma\,, \tag{4.26}$$

and

$$l_\Gamma(\boldsymbol{v}) = (\boldsymbol{f},\boldsymbol{v})_\Gamma. \tag{4.27}$$

Which is equivalent to

$$\begin{aligned}&\int_\Gamma 2\mu\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u}) : \boldsymbol{\varepsilon}_\Gamma(\boldsymbol{v})d\Gamma - \int_\Gamma 4\mu(\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u})\cdot\boldsymbol{n}) : (\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{v})\cdot\boldsymbol{n})d\Gamma \\ &+ \int_\Gamma \lambda_0(\nabla_\Gamma\cdot\boldsymbol{u})(\nabla_\Gamma\cdot\boldsymbol{v})d\Gamma = \int_\Gamma \boldsymbol{f}\boldsymbol{v}d\Gamma\end{aligned} \tag{4.28}$$

Using Mandel notation we have the strain tensor

$$\boldsymbol{\varepsilon}_\Gamma^M := \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \sqrt{2}\varepsilon_{12} \\ \sqrt{2}\varepsilon_{13} \\ \sqrt{2}\varepsilon_{23} \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial x_\Gamma} & 0 & 0 \\ 0 & \frac{\partial}{\partial y_\Gamma} & 0 \\ 0 & 0 & \frac{\partial}{\partial z_\Gamma} \\ \frac{1}{\sqrt{2}}\frac{\partial}{\partial x_\Gamma} & \frac{1}{\sqrt{2}}\frac{\partial}{\partial y_\Gamma} & 0 \\ \frac{1}{\sqrt{2}}\frac{\partial}{\partial x_\Gamma} & 0 & \frac{1}{\sqrt{2}}\frac{\partial}{\partial z_\Gamma} \\ 0 & \frac{1}{\sqrt{2}}\frac{\partial}{\partial y_\Gamma} & \frac{1}{\sqrt{2}}\frac{\partial}{\partial z_\Gamma} \end{pmatrix} \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}, \tag{4.29}$$

and using

$$\boldsymbol{\Phi} := \begin{pmatrix} \varphi_1 & 0 & 0 & \varphi_2 & 0 & 0 & \cdots \\ 0 & \varphi_1 & 0 & 0 & \varphi_2 & 0 & \cdots \\ 0 & 0 & \varphi_2 & 0 & 0 & \varphi_2 & \cdots \end{pmatrix}, \tag{4.30}$$

we get

$$\mathbf{B}_\varepsilon := \begin{pmatrix} \frac{\partial}{\partial x_\Gamma} & 0 & 0 \\ 0 & \frac{\partial}{\partial y_\Gamma} & 0 \\ 0 & 0 & \frac{\partial}{\partial z_\Gamma} \\ \frac{1}{\sqrt{2}}\frac{\partial}{\partial x_\Gamma} & \frac{1}{\sqrt{2}}\frac{\partial}{\partial y_\Gamma} & 0 \\ \frac{1}{\sqrt{2}}\frac{\partial}{\partial x_\Gamma} & 0 & \frac{1}{\sqrt{2}}\frac{\partial}{\partial z_\Gamma} \\ 0 & \frac{1}{\sqrt{2}}\frac{\partial}{\partial y_\Gamma} & \frac{1}{\sqrt{2}}\frac{\partial}{\partial z_\Gamma} \end{pmatrix} \boldsymbol{\Phi} \tag{4.31}$$

such that the Galerkin approximation of $\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u}) : \boldsymbol{\varepsilon}_\Gamma(\boldsymbol{v})$ yields $\mathbf{B}_\varepsilon^{\mathrm{T}}\mathbf{B}_\varepsilon\mathbf{U}$ and for the first term of (4.28)

$$\int_\Gamma 2\mu\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u}) : \boldsymbol{\varepsilon}_\Gamma(\boldsymbol{v})ds \approx \left(\int_\Gamma 2\mu\mathbf{B}_\varepsilon^{\mathrm{T}}\mathbf{B}_\varepsilon ds\right)\mathbf{U}. \tag{4.32}$$

For the second term of (4.28) we have that

$$\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u})\cdot\boldsymbol{n} = \begin{pmatrix} n_1\frac{\partial u_x}{\partial x} + n_2\frac{1}{2}\left(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}\right) + n_3\frac{1}{2}\left(\frac{\partial u_z}{\partial x} + \frac{\partial u_x}{\partial z}\right) \\ n_1\frac{1}{2}\left(\frac{\partial u_y}{\partial x} + \frac{\partial u_x}{\partial y}\right) + n_2\frac{\partial u_y}{\partial y} + n_3\frac{1}{2}\left(\frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y}\right) \\ n_1\frac{1}{2}\left(\frac{\partial u_z}{\partial x} + \frac{\partial u_x}{\partial z}\right) + n_2\frac{1}{2}\left(\frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y}\right) + n_3\frac{\partial u_z}{\partial z} \end{pmatrix}, \tag{4.33}$$

introducing the notation system

$$\boldsymbol{\varphi}_x^1 := \begin{pmatrix} \frac{\partial\varphi_1}{\partial x} & 0 & 0 & \frac{\partial\varphi_2}{\partial x} & 0 & 0 & \cdots \end{pmatrix} \tag{4.34}$$

$$\boldsymbol{\varphi}_x^2 := \begin{pmatrix} 0 & \frac{\partial\varphi_1}{\partial x} & 0 & 0 & \frac{\partial\varphi_2}{\partial x} & 0 & \cdots \end{pmatrix} \tag{4.35}$$

$$\boldsymbol{\varphi}_x^3 := \begin{pmatrix} 0 & 0 & \frac{\partial\varphi_1}{\partial x} & 0 & 0 & \frac{\partial\varphi_2}{\partial x} & \cdots \end{pmatrix} \tag{4.36}$$

$$\boldsymbol{\varphi}_y^1 := \begin{pmatrix} \frac{\partial\varphi_1}{\partial y} & 0 & 0 & \frac{\partial\varphi_2}{\partial y} & 0 & 0 & \cdots \end{pmatrix} \tag{4.37}$$

we have

$$\mathbf{B}_n = \begin{pmatrix} n_1\boldsymbol{\varphi}_x^1 + n_2\frac{1}{2}(\boldsymbol{\varphi}_y^1 + \boldsymbol{\varphi}_x^2) + n_3\frac{1}{2}(\boldsymbol{\varphi}_z^1 + \boldsymbol{\varphi}_x^3) \\ n_1\frac{1}{2}(\boldsymbol{\varphi}_y^1 + \boldsymbol{\varphi}_x^2) + n_2\boldsymbol{\varphi}_y^2 + n_3\frac{1}{2}(\boldsymbol{\varphi}_z^2 + \boldsymbol{\varphi}_y^3) \\ n_1\frac{1}{2}(\boldsymbol{\varphi}_z^1 + \boldsymbol{\varphi}_x^3) + n_2\frac{1}{2}(\boldsymbol{\varphi}_z^2 + \boldsymbol{\varphi}_y^3) + n_3\boldsymbol{\varphi}_z^3 \end{pmatrix} \tag{4.38}$$

and the second term approximation becomes

$$\int_\Gamma 4\mu(\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{u})\cdot\boldsymbol{n}) : (\boldsymbol{\varepsilon}_\Gamma(\boldsymbol{v})\cdot\boldsymbol{n})ds \approx \left(\int_\Gamma 4\mu\mathbf{B}_n^{\mathrm{T}}\mathbf{B}_n ds\right)\mathbf{U}. \tag{4.39}$$

For the third term we use the notation

$$\mathbf{B}_{div} := \begin{pmatrix} \frac{\partial\varphi_1}{\partial x} & \frac{\partial\varphi_1}{\partial y} & \frac{\partial\varphi_1}{\partial z} & \frac{\partial\varphi_2}{\partial x} & \frac{\partial\varphi_2}{\partial y} & \frac{\partial\varphi_2}{\partial z} & \cdots \end{pmatrix}, \tag{4.40}$$

and get

$$\int_\Gamma \lambda_0(\nabla_\Gamma\cdot\boldsymbol{u})(\nabla_\Gamma\cdot\boldsymbol{v})ds \approx \left(\int_\Gamma \lambda_0\mathbf{B}_{div}^{\mathrm{T}}\mathbf{B}_{div}ds\right)\mathbf{U}. \tag{4.41}$$

Finally we have the linear system

$$\left(\int_\Gamma 2\mu\mathbf{B}_\varepsilon^{\mathrm{T}}\mathbf{B}_\varepsilon ds - \int_\Gamma 4\mu\mathbf{B}_n^{\mathrm{T}}\mathbf{B}_n ds + \int_\Gamma \lambda_0\mathbf{B}_{div}^{\mathrm{T}}\mathbf{B}_{div}ds\right)\mathbf{U} = \int_\Gamma \boldsymbol{f}\boldsymbol{\Phi}ds \tag{4.42}$$

or

$$\mathbf{SU} = \mathbf{F} \tag{4.43}$$

4.4.2 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

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

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.

4.5 Method of manufactured solutions

The following illustrates an example case of generating a right-hand side to a PDE to be used in convergence analysis. Consider again the Laplace-Beltrami operator acting on a function $\boldsymbol{u}$. Let Γ be a smooth surface of the form of a torus with R and r being the major and minor radius of the torus. The Laplace-Beltrami equation on the surface Γ is given by

$$-\nabla_\Gamma \cdot (\nabla_\Gamma \boldsymbol{u}) = \boldsymbol{f} \tag{4.44}$$

In order to conduct error analysis on a computational model that solves (4.44) using the finite element method, we can assume a solution $\boldsymbol{u}$ and analytically work out the right hand side $\boldsymbol{f}$ in (4.44). We then plug in $\boldsymbol{f}$ in our computational model and compute the approximate solution $\boldsymbol{u}_h$. We then can compute the error $(\boldsymbol{u}_h - \boldsymbol{u})$ and analyze the convergence. We begin by assuming

$$\boldsymbol{u} = x + y + z, \tag{4.45}$$

and using the linear tangential operator $\boldsymbol{P}$ we get

$$-\nabla \cdot (\boldsymbol{P}\boldsymbol{P}\nabla u) = \boldsymbol{f}. \tag{4.46}$$

The gradient of the solution u is then

$$\nabla \boldsymbol{u} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}. \tag{4.47}$$

It is clear that the normal to the surface needs to be worked out analytically, i.e. we need to have $\boldsymbol{n} = \boldsymbol{n}(\boldsymbol{x})_\Gamma$. Defining the torus implicitly in Cartesian coordinates, radially symmetric about the z-axis

$$\phi(x, y, z) = \left(R - \sqrt{x^2 + y^2}\right)^2 + z^2 - r^2, \tag{4.48}$$

where $\phi(x, y, z) = 0$ is the solution to the isosurface Γ. Recalling that the gradient to a distance function ϕ coincides with the normal field $\boldsymbol{n}$ at $\phi = 0$, we have

$$\nabla\phi = \begin{pmatrix} x\left(2 - \dfrac{2R}{\sqrt{x^2 + y^2}}\right) \\ y\left(2 - \dfrac{2R}{\sqrt{x^2 + y^2}}\right) \\ 2z \end{pmatrix} \overset{\text{normalized}}{\Rightarrow} \begin{pmatrix} x\left(1 - \dfrac{R}{\sqrt{x^2 + y^2}}\right) \\ y\left(1 - \dfrac{R}{\sqrt{x^2 + y^2}}\right) \\ z \end{pmatrix} = \boldsymbol{n}. \tag{4.49}$$

Next we compute $\boldsymbol{PP}\nabla\boldsymbol{u}$

$$\hat{\boldsymbol{u}} = \boldsymbol{PP}\nabla\boldsymbol{u}, \tag{4.50}$$

and finally

$$\boldsymbol{f} = -\nabla \cdot \hat{\boldsymbol{u}} = -\frac{\partial}{\partial x}\hat{u_x} - \frac{\partial}{\partial y}\hat{u_y} - \frac{\partial}{\partial z}\hat{u_z}. \tag{4.51}$$

Working out (4.51) is best done in a symbolic processor such as Mathematica[2], see algorithm 4.1.

[2]https://www.wolfram.com/mathematica/

Algorithm 4.1 Working out the Laplace-Beltrami load on a torus in mathematica

Laplace-Beltrami load

```
In[139]:= u = x + y + z;

In[167]:= ∇[u]

Out[167]= {1, 1, 1}
```

In[168]:= $\phi = \left(R - \sqrt{x^2 + y^2}\right)^2 + z^2 - r^2;$

```
In[174]:= ∇[ϕ]
```

Out[174]=
$$\left\{-\frac{2\,x\left(R-\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}},\ -\frac{2\,y\left(R-\sqrt{x^2+y^2}\right)}{\sqrt{x^2+y^2}},\ 2\,z\right\}$$

```
In[181]:= n = ∇[ϕ] / 2 // Simplify
```

Out[181]=
$$\left\{x-\frac{R\,x}{\sqrt{x^2+y^2}},\ y-\frac{R\,y}{\sqrt{x^2+y^2}},\ z\right\}$$

```
In[182]:= P = 𝕀[3] - Outer[Times, n, n]
```

Out[182]=
$$\begin{pmatrix} 1-\left(x-\frac{R\,x}{\sqrt{x^2+y^2}}\right)^2 & -\left(x-\frac{R\,x}{\sqrt{x^2+y^2}}\right)\left(y-\frac{R\,y}{\sqrt{x^2+y^2}}\right) & -\left(x-\frac{R\,x}{\sqrt{x^2+y^2}}\right)z \\ -\left(x-\frac{R\,x}{\sqrt{x^2+y^2}}\right)\left(y-\frac{R\,y}{\sqrt{x^2+y^2}}\right) & 1-\left(y-\frac{R\,y}{\sqrt{x^2+y^2}}\right)^2 & -\left(y-\frac{R\,y}{\sqrt{x^2+y^2}}\right)z \\ -\left(x-\frac{R\,x}{\sqrt{x^2+y^2}}\right)z & -\left(y-\frac{R\,y}{\sqrt{x^2+y^2}}\right)z & 1-z^2 \end{pmatrix}$$

```
In[185]:= û = P.P.∇[u];

In[192]:= f = Div[-û, {x, y, z}] // Simplify
```

Out[192]=
$$\frac{1}{x^2+y^2}\Big(R^4\,(-(x+y)) + R^3\sqrt{x^2+y^2}\,(9\,x+9\,y+z) -$$
$$R^2\left(21\,x^3+x^2\,(21\,y+8\,z)+x\left(21\,y^2+z^2-2\right)+y\left(21\,y^2+8\,y\,z+z^2-2\right)\right)+$$
$$R\sqrt{x^2+y^2}\,\left(19\,x^3+x^2\,(19\,y+13\,z)+x\left(19\,y^2+7\,z^2-10\right)+19\,y^3+13\,y^2\,z+7\,y\,z^2-10\,y+z^3-2\,z\right)-$$
$$2\left(x^2+y^2\right)\left(3\,x^3+3\,x^2\,(y+z)+x\left(3\,y^2+3\,z^2-4\right)+3\,y^3+3\,y^2\,z+3\,y\,z^2-4\,y+3\,z^3-4\,z\right)\Big)$$

5 Summary of Appended Supplements

5.1 Supplement 1

A finite element method for finding minimal curvature on a surface based on computing a discrete Laplace-Beltrami operator that operates on the Cartesian coordinates of the surface is suggested. The surface is the discrete interface between a zero level set of a distance function and a linear tetrahedral mesh. The normal mean curvature flow is computed by solving the Laplace-Beltrami equation. The finite element discretization is done on the piecewise planar surface using the linear tetrahedral basis functions of the background mesh. Tangential calculus is employed and the tangential operator is used in the modeling of the Laplace-Beltrami operator. A FEM for computing the mean curvature vector needs stabilization and a recent stabilization method was successfully employed. In order to propagate the surface in the direction towards minimal curvature, a material time derivative of the distance function is discretized in time and space. Incremental updates of the distance function leads to a distortion such that the property of the function being a distance function is degraded, this needs to be addressed by reinitialization. Numerical experiments show good convergence with known analytical solutions. This method is suitable for evolving surface problems in which the surface is free to evolve into complicated shapes.

5.2 Supplement 2

A cut finite element method for the elastic membrane is suggested. Both free membrane and membranes coupled to 3D elasticity are considered. The membrane shell is modeled using tangential calculus embedded in a three dimensional space. The surface of the membrane is the discrete interface between a zero level set of a distance function and a three dimensional mesh of linear tetrahedra or tri-linear hexahedra. The mesh does not in general align with the surface of the membrane, which is instead allowed to arbitrarily cut the mesh. The finite element discretization is done on the piecewise planar surface using the basis functions of the background mesh. Due to the arbitrary cuts of the mesh, the size of the planar surface elements vary greatly in size. An integration over such surface elements leads, in the free membrane case, to severe ill conditioning of the stiffness matrix which needs to be stabilized. A stabilization method is proposed which provides stability to the solution and gives the right conditioning of the discrete system. This approach allows for rapid insertion of arbitrarily shaped membranes into already existing 3D finite element models. The suggested approach is numerically verified and gives errors comparable to triangulated membranes, using the same degree of approximation.

5.3 Supplement 3

In order to validate computational models by computing the error between the approximate solution and solutions from experimental data, one needs to have a proper framework for creating such a comparison. Let f^e denote a solution field from some experimental data of some load case. The corresponding approximate model solution field generated by a computational is then denoted $f^h(\boldsymbol{x})$, where $\boldsymbol{x}$ denotes a set of parameters chosen as the design variables of the computational model.

This is also known as the inverse problem of finding a set of model parameters that fit the observations. The validation is carried out considering a set of load cases, where each case yields a solution field f_i^e and a corresponding approximate solution $f_i^h(\boldsymbol{x})$ from the computational model. Next the error for each case is computed by carefully examining the solution fields and capturing the crucial parts of the fields. The L_2 error between the two solution fields is defined by

$$E_i := \beta||f_e^i - f_h^i||_{L_2}, \tag{5.1}$$

where $||\boldsymbol{u}-\boldsymbol{v}||_{L_2} = \sqrt{\int_\Omega(\boldsymbol{u}-\boldsymbol{v})^2 d\Omega}$ and β is used for penalization to achieve a better fit. We now can form an optimization problem of finding a set of computational parameters $\boldsymbol{x}$ that minimizes the errors E_i $i = 1, 2, ...$

$$\begin{cases} \min\limits_{\boldsymbol{x}} & E_i \quad i = 1, 2, ... \\ \text{subject to} & \begin{cases} g_i(\boldsymbol{x}) \leq 0 & i = 1, 2, ... \\ x_0^i \leq x^i & i = 1, 2, ... \\ x^i \leq x_1^i & i = 1, 2, ... \end{cases} \end{cases} \tag{5.2}$$

where g_i are a set of parameter dependent side conditions e.g. stress, displacement etc. The parameters x^i are usually bound by physical limitations.

Material parameters for a previously developed meso mechanical finite element model of a thin adhesive layer are optimized using the SPEA2 algorithm. Experimental data from previously performed experiments was compared to simulations in order to compute the L_2 error of two different load cases. From the error measures, two objectives were defined and used in a multi objective optimization study to generate a discrete Pareto set of optimal solutions. Compared to the original study where the two objectives were combined using a weighted sums approach and the resulting single objective optimization problem solved using an evolutionary algorithm, this study generated a Pareto front which can be used to verify the model, get insight into the correlation between different model parameters and provide good basis for the choice of parameters.

In the context of this thesis, the approach of this paper provides a solid framework for validation of computational models which are hard to validate without experimental data.

6 Future Work

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

For the elastic material, a possible application is to introduce anisotropy through two sets of fiber directions and optimize the amount and orientation of the fibers to maximize stiffness. This is under development at the time of writing.

A natural extension of Supplement 2 is to work out an hyper elastic model for large deformation elasticity. The challenge lies in the fact the deformation map is hard to define on implicit surfaces. This means that the deformation map is some sense needs to be a function of something else than the coordinates of the initial configuration.

Another idea is create anisotropic materials 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 has a configuration of discrete 1D fibers. This could be used as input data to create computational models for such composite objects.

Sandwich constructions is another topic that can be explored. Glue 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 discontinues Galerkin which makes it possible to introduce weakening and crack propagation in the models.

Structure optimization using cut finite element method, especially form finding is an area that is interesting. Shape finding traditionally involves meshing techniques which are not needed in the cutFEM setting.

Techniques relating to the cutFEM need further studies, for instance adaptivity to resolve the underlying mesh where the curvature of a surface is high is one area that requires attention.

Bibliography

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

[2] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon mesh processing*. CRC press, 2010.

[3] Erik Burman, Susanne Claus, Peter Hansbo, Mats G Larson, and André Massing. Cutfem: discretizing geometry and partial differential equations. *International Journal for Numerical Methods in Engineering*, 2014.

[4] Erik Burman, Peter Hansbo, and Mats 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.

[5] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim 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.

[6] Dominique Chapelle and Klaus-Jurgen Bathe. *The finite element analysis of shells-Fundamentals*. Springer Science & Business Media, 2010.

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

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

[9] François Cuvelier, Caroline Japhet, and Gilles Scarella. An efficient way to perform the assembly of finite element matrices in matlab and octave. *arXiv preprint arXiv:1305.3122*, 2013.

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

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

[12] Michel C Delfour and Jean-Paul Zolésio. Differential equations for linear shells: comparison between intrinsic and classical models. *Advances in mathematical sciences: CRMs*, 25:41–124, 1997.

[13] Gerhard Dziuk. *Finite elements for the Beltrami operator on arbitrary surfaces*. Springer, 1988.

[14] Gerhard Dziuk. An algorithm for evolutionary surfaces. *Numerische Mathematik*, 58(1):603–611, 1990.

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

[16] Peter Hansbo and Mats 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.

[17] Peter Hansbo, Mats G Larson, and Sara Zahedi. Stabilized finite element approximation of the mean curvature vector on closed surfaces. *arXiv preprint arXiv:1407.3043*, 2014.

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

[19] Gerhard A Holzapfel. *Nonlinear solid mechanics*, volume 24. Wiley Chichester, 2000.

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

[21] Stanley Osher and Ronald Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153. Springer Science & Business Media, 2006.

[22] James Albert 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.

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

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

Appended Papers

Supplement I	Mirza Cennanovic, Peter Hansbo, Mats G Larsson Minimal surface computation using finite element method on an embedded surface International Journal for Numerical Methods in Engineering, Wiley Online Library, 2015
Supplement II	Mirza Cennanovic, Peter Hansbo, Mats G Larsson Cut finite element modeling of linear membranes Submitted to Computer Methods in Applied Mechanics and Engineering
Supplement III	Kaveh Amouzgar, Mirza Cenanovic, Kent Salomonsson Multi-objective optimization of material model parameters of an adhesive layer by using SPEA2 11th World Congress on Structural and Multidisciplinary Optimization