NUMERICAL METHODS FOR PDES ON CURVES AND SURFACES

DIMITRIOS KAMILIS

Master of Science
Computational Science and Engineering

Umeå university

SUPERVISOR:
Mats G. Larson

DATE:
September 2013

ABSTRACT

Curves and surfaces are manifolds that can be represented using implicit and parametric methods. With a representation in hand, one can define a partial differential equation on the manifold using differential tangential calculus. The solution of these PDEs is quite interesting because they have many applications in a variety of areas including fluid dynamics, solid mechanics, biology and image processing.

In this thesis, we examine two numerical methods for the solution of PDEs on manifolds: a so called cut finite element method and isogeometric analysis. We review the theoretical framework of the two methods and implement them to solve example problems in two and three dimensions: the Laplace-Beltrami problem, the Laplace-Beltrami eigenvalue problem, the biharmonic problem and the time-dependent advection diffusion problem. We compare the methods and we confirm that the numerical results agree with the exact solutions and that they obey the theoretical a priori error estimates.

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor Professor Mats G. Larson for the introduction to this subject and for his comments throughout the project. I would also like to thank Professor Eddie Wadbro for his comments and corrections.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Partial differential equations (PDEs) on curves and surfaces appear in a variety of problems and applications in fluid dynamics, materials science, solid mechanics, biology and image processing. Examples are the modeling of interfaces in multiphase fluid flows and the modeling of surface active agents (surfactants) [JL04]. In this case, the equation on the interface is often coupled to the equation on the fixed bulk domain. Another example is in biology, where bio-membranes are treated as fluid surfaces [ES10]. In general, when the geometry of the physical problem can be considered "thin" in some direction, we can simplify the model using a formulation that involves PDEs on a lower dimensional geometry, i. e. on a curve or a surface. For example, this is a useful approximation when we want to model thin shells. PDEs on surfaces can also be used in image processing for shape recognition (shape DNA) [RWP06, RWSN09].

There are different ways to define and represent curves and surfaces [WRP07]. In the language of differential geometry [Bro08, Pre10, Ber03] they are manifolds (see A.1.1). Taking the extrinsic view, we can consider the curve or the surface as a lower dimensional manifold embedded in the physical space, namely as a hypersurface. In the parametric method, the manifold can be represented using a geometrical mapping from a parameter domain to the physical space. Another representation method is through an implicit function [OF03]. This method defines the interface as an isocontour or level set of some function. The interface is thus defined in one dimension lower than the implicit function. In contrast, in an explicit interface representation one defines explicitly the points which belong to the interface. Finally, curves and surfaces can also be represented using generative or procedural descriptions where points are generated through some process. Typical examples are subdivision schemes and fractals. Each representation method has advantages and disadvantages and is used according to the particular problem. In this thesis, we will only be concerned with the parametric representation through the use of B-splines and NURBS, and the implicit representation through the use of level set and signed distance functions. We will refer to curves and surfaces as hypersurfaces or manifolds in general, irrespective of the representation.

A very successful computational method for the solution of partial differential equations is the Finite Element Method (FEM) [LB12, BS08]. It has been studied extensively and has been ap-

plied to PDEs on manifolds using a variety of approaches. A review of finite element methods for the solution of partial differential equations on surfaces can be found in Dziuk and Elliott [DE13]. A first attempt for the solution of the Laplace-Beltrami problem on a curved surface was made by Dziuk [Dzi88] using triangulated surfaces and surface finite elements. Another approach has been the use of implicit level set functions to represent the hypersurface and solve the differential equation in a narrow band around it [DDEH09]. The method that we will follow is the one proposed by Olshanskii et al. [ORG09] which we will call a cut finite element method. The idea is that the hypersurface is embedded in a background domain and the finite element space defined on this domain induces a restricted finite element space on the manifold. A nice advantage of this method is that the same background grid can be used for both the implicit hypersurface representation and for the equation defined on it. Similarly, the same finite element space can be used for both the bulk domain and its interface.

In recent years, another computational method called Isogeometric Analysis (IGA) has been developed [HCB05, HCB09]. Its main goal has been to combine Computer Aided Design (CAD) and Finite Element Analysis (FEA) so as to integrate all tools in the design process. It uses computational geometry tools such as Non-Uniform Rational B-splines [PT97, Rog01] and it is based on the isoparametric paradigm which utilizes the same basis functions for both the parametric representation of the geometry and the approximation space used for the solution of the partial differential equation. In a sense, IGA is a superset of classical FEM. Advantages over FEM include the exact geometry representation and the arbitrary degree of continuity for the basis functions, while a disadvantage is the tensor product structure of NURBS which causes refinement to be a global operation. IGA provides a natural framework for the modeling and solution of partial differential equations on manifolds since the parametric nature of the basis functions and the isogeometric approach are well suited to these problems. In this thesis, we will follow the IGA approach for the solution of manifold PDEs as described in Dedè and Quarterioni [DQ13].

1.1 OBJECTIVE

The objective of this thesis is to review and examine finite element and isogeometric analysis methods for the solution of partial differential equations defined on curves and surfaces. This is achieved through the study of the theoretical frameworks and through the solution of example problems using an implementation of the two methods in MATLAB.

1.2 OUTLINE

The outline of this work is as follows:

CHAPTER 2 We introduce and review the parametric and implicit methods for the representation of curves and surfaces. We also introduce the basic concepts needed from differential geometry. Finally, we formulate example PDE problems on manifolds and obtain their weak form.

CHAPTER 3 We recall the basics of the Galerkin finite element method and we introduce the cut finite element method. We also provide implementation details.

CHAPTER 4 We introduce the basics of isogeometric analysis and we explain how it can be applied to solve partial differential equations on manifolds. We also describe implementation details.

CHAPTER 5 We report the numerical results for the different methods and example problems. We compare our results with the theoretical solutions and we plot the convergence rates as compared with the theoretical optimal error estimates. We also compare the performances of the finite element method and the isogeometric analysis method for the Laplace-Beltrami problem on a curve.

CHAPTER 6 We conclude our work and suggest areas of improvement.

APPENDIX A We provide some additional definitions and theorems for our theoretical framework.

APPENDIX B We list example programs from our implementation in MATLAB. This is not intended to provide an extensive listing of our code but it is rather intended to showcase the structure of our implementation and the most important code segments.

The implementation of the methods and the example problems was performed in MATLAB [MAT13]. In particular the implementation of the cut finite element method in 2D was developed entirely for the purposes of this thesis. For the implementation of the isogeometric analysis method, we extended the excellent GeoPDEs library [DFRV11] so as to facilitate the solution of PDEs on manifolds and to support the solution of additional problems.

2

CURVES AND SURFACES

We will consider curves and surfaces as manifolds embedded in the physical space, namely as hypersurfaces with codimension one. The basic definitions can be found in Appendix A and more details can be found in any Differential Geometry book. In this section, we will describe the implicit and parametric representation methods.

2.1 IMPLICIT REPRESENTATION

The implicit representation of curves and surfaces has proved to be versatile and efficient in a variety of applications in computer graphics. Complicated and evolving hypersurfaces can be handled with simple algorithms within the implicit approach. The method might at first be seen as wasteful since the implicit function is defined in a space that is one dimension higher than the interface. However, it turns out that this has advantages when, for example, one wants to model an evolving interface.

Definition 2.1.1. *An implicit representation of a n-dimensional hypersurface Σ takes the form*

$$\Sigma = \{\mathbf{x} \in \mathbb{R}^{n+1} \mid \phi(\mathbf{x}) = 0\}, \tag{2.1.1}$$

where $\mathbf{x}$ is a point on the hypersurface implicitly defined by the function $\phi : \mathbb{R}^{n+1} \to \mathbb{R}$.

The representation embeds the interface in a domain of one dimension higher and defines it as the zero level or isocontour of the implicit function. For example, a unit sphere (a 2D surface embedded in $\mathbb{R}^3$) can be written as

$$\phi(\mathbf{x}) = \|\mathbf{x}\|^2 - 1 = 0\,, \text{ where } \|\mathbf{x}\|^2 = x^2 + y^2 + z^2. \tag{2.1.2}$$

The implicit function has the property

$$\begin{aligned} &\phi(\mathbf{x}) > 0\,, \text{ if } \mathbf{x} \text{ is "outside" the hypersurface} \\ &\phi(\mathbf{x}) < 0\,, \text{ if } \mathbf{x} \text{ is "inside" the hypersurface} \\ &\phi(\mathbf{x}) = 0\,, \text{ if } \mathbf{x} \text{ is exactly on the hypersurface} \end{aligned} \tag{2.1.3}$$

This property allows one to easily test whether one point lies inside or outside the interface.

The gradient of the implicit function is perpendicular to the isocontours and it points in the direction of increasing ϕ. In general, we desire that the implicit function is smooth and hence

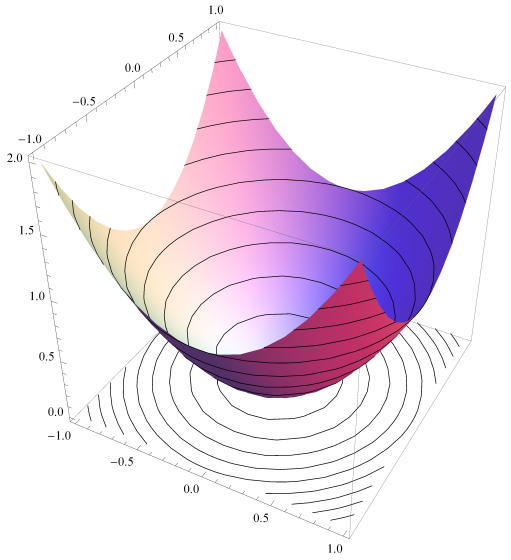

Figure 1: Example of the implicit representation. Circles of different radius defined as level sets of an implicit function.

its gradient is well defined everywhere. We define the normal vector field by

$$\mathbf{n}(\mathbf{x}) := \frac{\nabla\phi(\mathbf{x})}{\|\nabla\phi(\mathbf{x})\|} \quad , \mathbf{x} \in \mathbb{R}^{n+1}. \tag{2.1.4}$$

The normal vector field is equal to the outward unit normal $\mathbf{n}_\Sigma$ for points on the hypersurface. Note that it can be undefined at specific points where the denominator is zero but this problem can be solved by arbitrarily assigning a value at these points.

2.1.1 *Signed distance functions*

Definition 2.1.2. *A distance function* $d(\mathbf{x})$ *is defined as*

$$d(\mathbf{x}) = \min(\|\mathbf{x} - \mathbf{x}_I\|) \text{ , for all } \mathbf{x}_I \in \Sigma, \tag{2.1.5}$$

which means that $d(\mathbf{x}) = 0$ *on* Σ.

A distance function obeys the property

$$\|\nabla d\| = 1. \tag{2.1.6}$$

Remark 2.1.1. *Property* (2.1.6) *is satisfied only in a general sense, since it is not true for points that are equidistant from at least two points on the interface. However, equations that are true in a general sense can still be used in numerical calculations as long as they do not destroy the numerical method entirely when they fail.*

Definition 2.1.3. *A signed or oriented distance function is an implicit function* ϕ_d *such that* $|\phi_d(\mathbf{x})| = d(\mathbf{x})$ *for all* $\mathbf{x}$.

An example of a signed distance function representation of the unit sphere is

$$\phi_d(\mathbf{x}) = \|\mathbf{x}\| - 1 = 0. \tag{2.1.7}$$

Note that property (2.1.6) is true for signed distance functions too.

The signed distance function enables us to find the closest point $\eta(\mathbf{x})$ on Σ to a point $\mathbf{x} \in \mathbb{R}^{n+1}$ by taking

$$\boldsymbol{\eta}(\mathbf{x}) = \mathbf{x} - \phi_d(\mathbf{x})\mathbf{n}(\mathbf{x}). \tag{2.1.8}$$

To make the closest point mapping unique, we define a volumetric neighborhood U (e.g. a tubular region) around Σ

$$U = \{\mathbf{x} \in \mathbb{R}^{n+1} \mid d(\mathbf{x}) < \delta\}, \tag{2.1.9}$$

where δ is small enough to guarantee uniqueness of the mapping $\boldsymbol{\eta} : U \to \Sigma$. The closest point mapping allows us to introduce a global coordinate system around Σ as for every point $\mathbf{x} \in U$ we can assign the *Fermi coordinates* $\phi_d(\mathbf{x})$ and $\boldsymbol{\eta}(\mathbf{x})$. Note that $\mathbf{n}(\mathbf{x}) = \mathbf{n}(\boldsymbol{\eta}(\mathbf{x}))$ for every point $\mathbf{x} \in U$. Also, any function f defined on Σ can be extended to U as $f^e(\mathbf{x}) = f(\boldsymbol{\eta}(\mathbf{x}))$.

2.2 PARAMETRIC REPRESENTATION

In a parametric representation the coordinates of a point on the hypersurface are represented separately as functions of independent parameters[1] through the use of geometrical mappings.

Definition 2.2.1. *In the parametric representation we assume that the hypersurface $\Sigma \subset \mathbb{R}^n$ is a Riemannian manifold[2] embedded in the physical space $\mathbb{R}^n$ and obtained through a geometrical mapping*

$$\mathbf{x} : \widehat{\Sigma} \to \Sigma, \tag{2.2.1}$$

where $\widehat{\Sigma} \subset \mathbb{R}^\kappa$ is the parameter space such that $n > \kappa \geqslant 1$.

The mapping is assumed to be smooth with piecewise smooth inverse. An example is a surface in $\mathbb{R}^3$ with $n = 3$ and $\kappa = 2$. The parameter space is defined through independent parameters $\boldsymbol{\xi} = (\xi_1, \cdots, \xi_\kappa) \in \mathbb{R}^\kappa$ while for the natural space we use Cartesian coordinates $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ so that the mapping (2.2.1) is equivalently expressed as $\boldsymbol{\xi} \to \mathbf{x}(\boldsymbol{\xi})$. From now on we will denote quantities that are defined in the parameter space using a hat. For example, we can define the Jacobian of the mapping $\widehat{J} : \widehat{\Sigma} \to \mathbb{R}^{n\times\kappa}$ as:

$$\widehat{J}_{ij}(\boldsymbol{\xi}) := \frac{\partial x_i}{\partial \xi_j}(\boldsymbol{\xi}) \ , \ i = 1, \ldots, n \quad j = 1, \ldots, \kappa. \tag{2.2.2}$$

1 The parameters are often chosen to lie in the unit interval.

2 We assume that the manifold is smooth, compact, connected, and oriented.

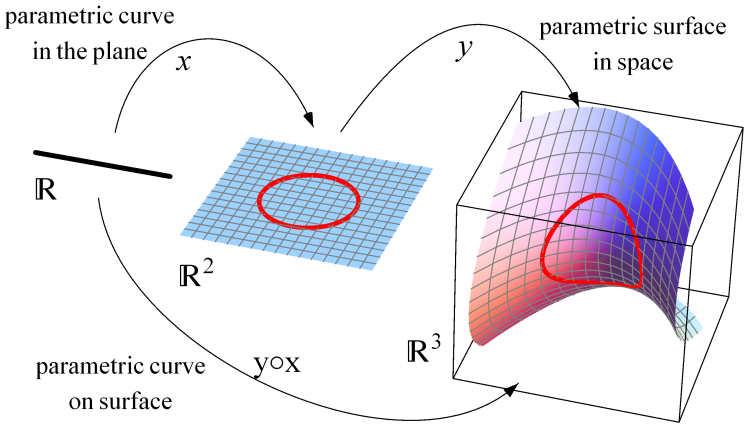

Figure 2: Examples of parametric representation: a parametric curve in 2D, a parametric curve in 3D and a parametric surface in 3D.

We can then define the first fundamental form (the induced metric on the manifold) $\widehat{G} : \widehat{\Sigma} \to \mathbb{R}^{\kappa \times \kappa}$ as:

$$\widehat{G}(\xi) := \widehat{J}^T \widehat{J}, \tag{2.2.3}$$

and the square root of the determinant $\widehat{g} : \widehat{\Sigma} \to \mathbb{R}$ as:

$$\widehat{g}(\xi) := \sqrt{\det(\widehat{G})}. \tag{2.2.4}$$

A function $\phi(\mathbf{x})$ defined on the hypersurface Σ can be pulled back to the parameter domain using the geometrical mapping so that

$$\widehat{\phi}(\xi) = \phi(\mathbf{x}(\xi)). \tag{2.2.5}$$

We can then also define the push-forward

$$\phi(\mathbf{x}) = \widehat{\phi}(\xi) \circ \mathbf{x}^{-1}(\xi), \tag{2.2.6}$$

and drop the distinction between the function definition on the parameter domain and the definition on the physical space.

2.3 COMPARISON OF IMPLICIT AND PARAMETRIC METHODS

A basic comparison between the implicit and parametric representations reveals some of the differences:

- The parametric representation allows for a natural direction of traversal while the implicit form does not.
- It is easier to generate points in the parametric form. This is an advantage for example in rendering.

- Given a point it is difficult to determine whether it is on the hypersurface when using the parametric form. It is naturally easy in the implicit form.
- The parametric representation can produce singularities which are not present in the actual geometry such as the poles.
- The implicit representation allows for the efficient application of Boolean operations, topology changes and surface intersection.

2.4 DIFFERENTIAL TANGENTIAL CALCULUS

We can define the hypersurface gradient as the projection of the physical space gradient of an extension of the function onto the tangent space of the hypersurface. More precisely:

Definition 2.4.1. *Let $\phi : \Sigma \to \mathbb{R}$ be a function on the hypersurface Σ. We denote by $\tilde{\phi}$ the smooth extension of ϕ to a neighborhood U of Σ, so that $\tilde{\phi}|_\Sigma = \phi$.*

An extension can be achieved using the closest point mapping (2.1.8) together with the uniqueness requirement of (2.1.9). In this case we can define $\tilde{\phi}(\mathbf{x}) = \phi \circ \mathbf{\eta}(\mathbf{x})$ for a point $\mathbf{x} \in \mathbb{R}^{n+1}$.

Definition 2.4.2. *For each $\mathbf{x} \in U$, the tangential projector operator $\mathbf{P}(\mathbf{x}) \in \mathbb{R}^{(n+1)\times(n+1)}$ is*

$$\mathbf{P}(\mathbf{x}) := \mathbf{I} - \mathbf{n}(\mathbf{x}) \otimes \mathbf{n}(\mathbf{x}), \tag{2.4.1}$$

where $\mathbf{I}$ is the identity tensor and $\mathbf{n}(\mathbf{x})$ is the unit normal field.

Definition 2.4.3. *The tangential gradient of a $C^1(\Sigma)$ function ϕ at $\mathbf{x} \in \Sigma$ is defined by*

$$\nabla_\Sigma \phi(\mathbf{x}) := \mathbf{P}(\mathbf{x}) \nabla \tilde{\phi}(\mathbf{x}), \tag{2.4.2}$$

where ∇ represents the usual gradient operator in $\mathbb{R}^{n+1}$.

The tangential gradient depends only on the values of $\tilde{\phi}$ restricted to the hypersurface and we also have that $\nabla_\Sigma \phi(\mathbf{x}) \cdot \mathbf{n}_\Sigma = 0$ so that the tangential gradient "lives" on the tangent space $T_x\Sigma$ at each point $\mathbf{x} \in \Sigma$. For a vector valued function $\mathbf{v}(\mathbf{x})$, the tangential divergence is defined as

$$\nabla_\Sigma \cdot \mathbf{v}(\mathbf{x}) = \operatorname{tr}(\nabla_\Sigma \mathbf{v}(\mathbf{x})). \tag{2.4.3}$$

Finally, the mean curvature[3] of Σ with respect to the outward pointing unit normal $\mathbf{n}_\Sigma$ is defined as

$$H = -\nabla_\Sigma \cdot \mathbf{n}_\Sigma. \tag{2.4.4}$$

3 We define the mean curvature as the sum of the principal curvatures rather than the arithmetic mean.

Definition 2.4.4. *The Laplace-Beltrami operator for a function* $\phi \in C^2(\Sigma)$ *is defined as*

$$\Delta_\Sigma \phi(\mathbf{x}) := \nabla_\Sigma \cdot \nabla_\Sigma \phi(\mathbf{x}), \tag{2.4.5}$$

so it is a generalization of the Laplace operator for curved geometries and it takes the usual form in flat Euclidean space.

2.4.1 *Differential operators in parameter space*

Using the geometrical mapping of the parametric representation we can rewrite differential operators using quantities in the parameter space. We start by writing the gradient in the physical space as

$$\nabla \tilde{\phi}(\mathbf{x}) = \left[\widehat{J}(\xi)\, \widehat{G}^{-1}(\xi) \widehat{\nabla}\widehat{\phi}(\xi)\right] \circ \mathbf{x}^{-1}(\xi), \tag{2.4.6}$$

where $\widehat{\nabla}\widehat{\phi}(\xi)$ denotes the gradient in the parameter space. The tangential gradient becomes

$$\begin{aligned} \nabla_\Sigma \phi(\mathbf{x}) &= \nabla \tilde{\phi}(\mathbf{x}) - \cancel{(\mathbf{n} \otimes \mathbf{n}) \nabla \tilde{\phi}(\mathbf{x})} \\ &= \left[\widehat{J}(\xi)\, \widehat{G}^{-1}(\xi) \widehat{\nabla}\widehat{\phi}(\xi)\right] \circ \mathbf{x}^{-1}(\xi). \end{aligned} \tag{2.4.7}$$

The fact that the second term is zero can easily be seen in the case of a curve in 2D space, where $\mathbf{x} = (x, y)$ and $\xi = \xi$. We have

$$\mathbf{t} = \frac{\partial \mathbf{x}}{\partial \xi} \quad , \quad \widehat{J} = \begin{pmatrix} \frac{\partial x}{\partial \xi} \\ \frac{\partial y}{\partial \xi} \end{pmatrix} = \mathbf{t}, \tag{2.4.8}$$

with $\mathbf{t}$ being the tangent vector and $\mathbf{n} \perp \mathbf{t}$. So

$$(\mathbf{n} \otimes \mathbf{n}) \nabla \tilde{\phi}(\mathbf{x}) = \left[(\mathbf{n}\mathbf{n}^T\, \mathbf{t})\, \widehat{G}^{-1}(\xi) \widehat{\nabla}\widehat{\phi}(\xi)\right] \circ \mathbf{x}^{-1}(\xi) \tag{2.4.9}$$

$$= 0. \tag{2.4.10}$$

Similarly the Laplace-Beltrami operator can be written in terms of parameter space quantities as [Ber03]

$$\Delta_\Sigma \phi(\mathbf{x}) = \left[\frac{1}{\widehat{g}(\xi)} \widehat{\nabla} \cdot \left(\widehat{g}(\xi) \widehat{G}^{-1}(\xi) \widehat{\nabla}\widehat{\phi}(\xi)\right)\right] \circ \mathbf{x}^{-1}(\xi). \tag{2.4.11}$$

Also the differentials used in integrals are transformed as

$$d\mathbf{x} = \hat{g}\, d\xi. \tag{2.4.12}$$

and any field f defined on Σ can be transformed to the parameter space as

$$\hat{f}(\xi) = f(\mathbf{x}(\xi)). \tag{2.4.13}$$

2.5 PDES ON CURVES AND SURFACES

Having defined tangential differential operators we can now formulate example PDEs on the manifold. If the manifold has a boundary $\Gamma = \partial\Sigma$ we can split it into two non-overlapping domains such that $\Gamma = \Gamma_D \cup \Gamma_N$ and $\Gamma_D \cap \Gamma_N = \emptyset$. We can then apply Dirichlet boundary conditions on Γ_D and Neumann boundary conditions on Γ_N. For simplicity we will only consider homogeneous boundary conditions. Some basic definitions from functional analysis can be found in Appendix A.

2.5.1 *The Laplace-Beltrami problem*

The Laplace-Beltrami problem is similar to the Poisson problem and takes the strong form: Given a source function $f \in L^2(\Sigma)$, find $u : \Sigma \to \mathbb{R}$ such that

$$-\Delta_\Sigma u = f \quad \text{on } \Sigma, \tag{2.5.1a}$$

$$u = 0 \quad \text{on } \Gamma_D, \tag{2.5.1b}$$

$$\mathbf{n}_\Gamma \cdot \nabla_\Sigma u = 0 \quad \text{on } \Gamma_N, \tag{2.5.1c}$$

where $\mathbf{n}_\Gamma$ is the unit normal on the boundary. For manifolds without boundary we need an additional condition for the problem to be well-posed. The reason is that we can choose $u = \text{const.}$ which makes the left hand side of (2.5.1a) to vanish. Since the null space of the Laplace-Beltrami operator on a closed manifold is the space of constant functions, we need to add an additional constraint to only allow the zero function. This zero mean constraint is

$$\int_\Sigma u \, d\mathbf{x} = 0. \tag{2.5.2}$$

2.5.1.1 *Variational form*

In variational or weak form we multiply the strong form with a test function $v \in V$ where V is a suitable function space. Then we integrate over the domain and use an appropriate form of Green's formula to rewrite the equation. For the tangential operators, Green's formula becomes:

$$(\nabla_\Sigma \cdot \boldsymbol{w}, v)_\Sigma = (\mathbf{n}_\Gamma \cdot \boldsymbol{w}, v)_\Gamma - (\boldsymbol{w}, \nabla_\Sigma v)_\Sigma + (\boldsymbol{w}, H\mathbf{n}_\Sigma v)_\Sigma, \tag{2.5.3}$$

where H is the mean curvature and $\mathbf{n}$ the outward pointing unit normal to the hypersurface.

In the case of the Laplace-Beltrami problem we apply Green's formula with $\boldsymbol{w} = \nabla_\Sigma u$ and we get:

$$\begin{aligned} -(\Delta_\Sigma u, v)_\Sigma &= (\nabla_\Sigma u, \nabla_\Sigma v)_\Sigma - (\mathbf{n}_\Gamma \cdot \nabla_\Sigma u, v)_{\Gamma_N} \\ &\quad - (\mathbf{n}_\Gamma \cdot \nabla_\Sigma u, v)_{\Gamma_D} - (\nabla_\Sigma u, (-\nabla_\Sigma \cdot \mathbf{n}_\Sigma)\mathbf{n}_\Sigma\, v)_\Sigma \\ &= (f, v)_\Sigma. \end{aligned} \tag{2.5.4}$$

The curvature term is zero because $\nabla_\Sigma v \cdot \mathbf{n}_\Sigma = 0$. The boundary terms are zero for manifolds without boundary. For manifolds with boundary, the boundary term for Γ_N is zero due to homogeneous Neumann conditions and the boundary term for Γ_D is zero due to the choice of the test function space. In particular, the weak form of the Laplace-Beltrami problem is: find $u \in V$ such that

$$a(u, v) = f(v) \quad , \forall v \in V_0, \tag{2.5.5}$$

where $a(u, v) = (\nabla_\Sigma u, \nabla_\Sigma v)_\Sigma$ is the symmetric bilinear form and $f(v) = (f, v)_\Sigma$ is the linear functional form associated with the problem. For manifolds with boundary the trial and test function spaces are equal so $V = V_0 = \{v \in H^1(\Sigma) : v|_{\Gamma_D} = 0\}$. For manifolds without boundary, the trial function space is instead $V = \{v \in H^1(\Sigma) : \int_\Sigma v \, d\mathbf{x} = 0\}$.

2.5.2 *The Laplace-Beltrami eigenvalue problem*

The Laplace-Beltrami eigenvalue problem takes the form: Find u and λ such that

$$-\Delta_\Sigma u = \lambda u \quad \text{on } \Sigma. \tag{2.5.6}$$

Similarly to the Laplace-Beltrami problem, we can consider homogeneous Dirichlet, homogeneous Neumann or no-boundary conditions. The Laplace-Beltrami eigenvalue problem becomes in weak form: find $u \in V$ and $\lambda \in \mathbb{R}$ such that

$$a(u, v) = \lambda m(u, v) \quad , \forall v \in V_0, \tag{2.5.7}$$

where $m(u, v) = (u, v)_\Sigma$ is a symmetric bilinear form. The choice of spaces V, V_0 is similar as in the Laplace-Beltrami problem but there is no need for the constraint (2.5.2) when the manifold is closed since the first eigenvalue is zero in this case. Since the problem is symmetric, the eigenvalues λ are real and form a diverging sequence $0 \leqslant \lambda_1 \leqslant \lambda_2 \leqslant \cdots \uparrow +\infty$ with each eigenvalue repeated according to its multiplicity. The spectrum of the Laplace-Beltrami operator has been the subject of analysis and the base of a method to compare and analyze different shapes (shape DNA) [RWP06, RWSN09].

2.5.3 *The biharmonic problem*

The biharmonic problem is a higher-order (fourth order) PDE problem and with Dirichlet boundary conditions it is defined as: given $f \in L^2(\Sigma)$ find $u : \Sigma \to \mathbb{R}$ such that

$$\Delta_\Sigma^2 u = f \quad \text{on } \Sigma, \tag{2.5.8a}$$
$$u = 0 \quad \text{on } \Gamma, \tag{2.5.8b}$$
$$\mathbf{n}_\Gamma \cdot \nabla_\Sigma u = 0 \quad \text{on } \Gamma, \tag{2.5.8c}$$

where $\Delta_\Sigma^2 u = \Delta_\Sigma(\Delta_\Sigma u)$. If the manifold has no boundary, we again need to impose the constraint (2.5.2). In weak form we apply Green's formula (2.5.3) twice and the problem becomes:

$$\begin{aligned}(\Delta_\Sigma^2 u, v)_\Sigma &= (\mathbf{n}_\Gamma \cdot \nabla_\Sigma(\Delta_\Sigma u), v)_\Gamma - (\nabla_\Sigma(\Delta_\Sigma u), \nabla_\Sigma v)_\Sigma \\ &\quad + \cancel{(\nabla_\Sigma(\Delta_\Sigma u), H\mathbf{n}_\Sigma v)_\Sigma} \\ &= (\mathbf{n}_\Gamma \cdot \nabla_\Sigma(\Delta_\Sigma u), v)_\Gamma - (\Delta_\Sigma u, \mathbf{n}_\Gamma \cdot \nabla_\Sigma v)_\Gamma \\ &\quad + (\Delta_\Sigma u, \Delta_\Sigma v)_\Sigma - \cancel{(H\mathbf{n}_\Sigma \Delta_\Sigma u, \nabla_\Sigma v)_\Sigma}.\end{aligned} \tag{2.5.9}$$

The boundary terms vanish by choosing the test space as $V_0 = \{v \in H^2(\Sigma) : v|_\Gamma = 0\,,\ (\mathbf{n}_\Gamma \cdot \nabla_\Sigma u)|_\Gamma = 0\}$. So the weak problem is: find $u \in V$ such that

$$a(u, v) = f(v) \quad , \forall v \in V_0, \tag{2.5.10}$$

where $a(u, v) = (\Delta_\Sigma u, \Delta_\Sigma v)_\Sigma$ and $f(v) = (f, v)_\Sigma$. The trial space is $V = V_0$ for manifolds with boundary and $V = \{v \in H^2(\Sigma) : \int_\Sigma v \, d\mathbf{x} = 0\}$ for closed manifolds.

2.5.4 *The time-dependent advection-diffusion problem*

The time-dependent advection-diffusion problem takes the form: given $f \in L^2(\Sigma)$ find $u(t) : \Sigma \times (0, T) \to \mathbb{R}$ such that

$$\frac{\partial u}{\partial t}(t) - \mu \Delta_\Sigma u(t) + \mathbf{b} \cdot \nabla_\Sigma u(t) = f \qquad \text{on } \Sigma \times (0, T), \tag{2.5.11a}$$
$$u(t) = 0 \qquad \text{on } \Gamma_D \times (0, T), \tag{2.5.11b}$$
$$\mathbf{n}_\Gamma \cdot \mu \nabla_\Sigma u(t) = 0 \qquad \text{on } \Gamma_N \times (0, T), \tag{2.5.11c}$$
$$u(0) = u_0 \qquad \text{on } \Sigma \times \{0\}, \tag{2.5.11d}$$

where $\mu \in (0, 1)$ is the diffusion parameter that controls the strength of the diffusion term and $\mathbf{b}$ is the advection field such that $\mathbf{b} \cdot \mathbf{n}_\Sigma = 0$, $\nabla_\Sigma \cdot \mathbf{b} = 0$. The weak form becomes: find $u \in V$

with $u(0) = u_0$ such that

$$m(\partial_t u, v) + \beta(u, v) + \alpha(u, v) = f(v) \quad , \forall t \in (0, T) \, , \forall v \in V_0, \tag{2.5.12}$$

where $\alpha(u, v) = (\mu \nabla_\Sigma u, \nabla_\Sigma v)_\Sigma$, $\beta(u, v) = (\mathbf{b} \cdot \nabla u, v)_\Sigma$ and $m(u, v) = (u, v)_\Sigma$ are the bilinear forms. The trial and test function spaces are $V = V_0 = \{v \in H^1(\Sigma) \, : \, v|_\Gamma = 0\}$.

When the advection-diffusion problem is advection dominated i.e. when μ is small, the solution exhibits regions where it changes rapidly. These so called layers may trigger oscillations throughout the computational domain and "destroy" the approximation. The onset of these oscillations is a mesh resolution problem and it happens when the diffusion parameter μ is smaller than the mesh size h. A stabilization method is needed in these cases to counteract this behavior. There are some possible choices here, for example the Galerkin Least Squares (GLS) stabilization and the Streamline Upwind Petrov Galerkin (SUPG) stabilization. The idea is to augment the weak form (2.5.12) with a stabilization term on the left hand side. We will use SUPG stabilization when solving the time-dependent advection-diffusion problem with isogeometric analysis.

2.5.5 *Existence and uniqueness of solutions*

For all the problems we described, existence and uniqueness of solutions are guaranteed by the Lax-Milgram lemma which states that:

Lax-Milgram lemma. *Let V be a Hilbert space with inner product $(\cdot, \cdot)$, a coercive continuous bilinear form $\alpha(\cdot, \cdot)$ on V and a continuous linear form $l(\cdot)$ on V. Then, there exists a unique solution $u \in V$ to the variational problem: find $u \in V$ such that*

$$\alpha(u, v) = l(v) \quad , \forall v \in V, \tag{2.5.13}$$

where coercivity of the bilinear form is defined as: there is a constant m such that

$$m\|v\|_V^2 \leqslant \alpha(u, v) \quad , \forall v \in V, \tag{2.5.14}$$

and continuity of the bilinear form is defined as: there is a constant C_α such that

$$\alpha(u, v) \leqslant C_\alpha \|u\|_V \|v\|_V \quad , \forall u, v \in V. \tag{2.5.15}$$

Finally, continuity of the linear form is defined as: there is a constant C_l such that

$$l(v) \leqslant C_l \|v\|_V \quad , \forall v \in V. \tag{2.5.16}$$

More details can be found in any Finite Element Method textbook [BS08, LB12]. Here we will use the Lax-Milgram lemma to prove the uniqueness and existence of the solution for the Laplace-Beltrami problem. We start by noting that for this problem the norm on the vector space V is

$$\|u\|_V^2 = \|\nabla_\Sigma u\|_{L^2(\Sigma)}^2 = (\nabla_\Sigma u, \nabla_\Sigma u)_{L^2(\Sigma)}. \tag{2.5.17}$$

Then we prove the coercivity of $\alpha(\cdot,\cdot)$

$$\alpha(u,u) = (\nabla_\Sigma u, \nabla_\Sigma u)_\Sigma \geqslant m\|u\|_V^2, \tag{2.5.18}$$

with $m = 1$. Next we prove the continuity of $\alpha(\cdot,\cdot)$ using the Cauchy-Schwarz inequality

$$\begin{aligned}\alpha(u,v) = (\nabla_\Sigma u, \nabla_\Sigma v)_\Sigma &\leqslant \|\nabla_\Sigma u\|_{L^2(\Sigma)}\|\nabla_\Sigma v\|_{L^2(\Sigma)} \\ &\leqslant \|u\|_V\|v\|_V.\end{aligned} \tag{2.5.19}$$

To prove the continuity of $l(\cdot)$ we need the Poincaré inequality which states that

$$\|u\|_{L^2(\Sigma)} \leqslant C\|\nabla_\Sigma u\|_{L^2(\Sigma)}. \tag{2.5.20}$$

Then we can prove the continuity of $l(\cdot)$ using the Cauchy-Schwarz inequality and the Poincaré inequality

$$\begin{aligned}l(v) = (f,v)_{L^2(\Sigma)} &\leqslant \|f\|_{L^2(\Sigma)}\|v\|_{L^2(\Sigma)} \\ &\leqslant C\|f\|_{L^2(\Sigma)}\|\nabla_\Sigma v\|_{L^2(\Sigma)} \\ &\leqslant C\|f\|_{L^2(\Sigma)}\|v\|_V = C\|v\|_V.\end{aligned} \tag{2.5.21}$$

The requirements of the Lax-Milgram lemma are satisfied, hence the solution to the Laplace-Beltrami problem exists and is unique.

3

FINITE ELEMENTS FOR PDES ON MANIFOLDS

As we described in the introduction, there is a variety of methods for the solution of PDEs on manifolds using the finite element method. We will focus on the cut finite element method for hypersurface PDEs as introduced by Olshanskii et al. [ORG09]. The matrix properties of this method have been examined in detail [OR10] and the method was further explored in subsequent papers [ORX12, ORX13c]. Moreover, it was applied to advection-diffusion problems [ORX13a], the biharmonic problem [LL13] and to diffusion problems for evolving hypersurfaces [HLZ13, ORX13b]. The method uses the implicit representation of the hypersurface and the finite element method to obtain an approximate solution. We begin by recalling the Galerkin finite element method.

3.1 GALERKIN FINITE ELEMENT METHOD

After obtaining the variational form of the PDE problem, the Galerkin finite element method substitutes the function space V with a finite dimensional subspace $V_h \subset V$ which in the case of the classical finite element method consists of continuous piecewise polynomials on a mesh K. Therefore the finite element solution becomes: find $u_h \in V_h$ such that

$$a(u_h, v) = l(v) \quad , \forall v \in V_h. \tag{3.1.1}$$

Using a basis $\{\phi_i\}_{i=1}^n$ for V_h the finite element solution can be written as

$$u_h = \sum_{i=1}^{n} \xi_i \phi_i, \tag{3.1.2}$$

where ξ_i are coefficients (the problem unknowns). The finite element method gives a linear system

$$A\xi = b, \tag{3.1.3}$$

where A is the stiffness matrix with entries $A_{ij} = a(\phi_i, \phi_j)$ and b is the load vector with entries $b_i = l(\phi_i)$.

3.1.1 *A priori error estimates*

The finite element approximation satisfies a priori estimates which express the error in terms of the exact solution. For example, the

best approximation result states that the error $e = u - u_h$ satisfies

$$\|e\|_V \leqslant \frac{C_\alpha}{m}\|u - v\|_V \quad , \forall v \in V_h. \tag{3.1.4}$$

This can be further quantified using interpolation estimates to produce optimal error bounds.

3.2 THE CUT FINITE ELEMENT METHOD

The cut finite element method for hypersurface PDEs consists of a finite element (A.3.2) discretization using an approximation Σ_h of the hypersurface[1] Σ and a restriction of the outer finite element spaces. We first consider a fixed domain $\Omega_I \subset \mathbb{R}^n$ that completely contains Σ with n being the dimension of the embedding physical space. We then assume there is a partition $\mathcal{T}_h = \{S\}$ of the domain Ω_I which consists of regular polygons e.g. regular quadrilaterals for $n = 2$ or regular tetrahedra for $n = 3$. We assume a discretization Σ_h of Σ that is consistent with the background domain partitioning and we define the set of elements $\mathcal{K}_h$ than intersect Σ_h by

$$\mathcal{K}_h = \{S \in \mathcal{T}_h \,:\, S \cap \Sigma_h \neq \emptyset\}. \tag{3.2.1}$$

The union of all elements in $\mathcal{K}_h$ forms the domain $\Omega_h = \cup_{S\in\mathcal{K}_h} S$. For each element $S \in \Omega_h$ we denote by T the intersection of the element with Σ_h so that $T = S \cap \Sigma_h$.

Remark 3.2.1. *In general T can coincide with the face of an element in $\mathcal{T}_h$ and thus the corresponding S is not unique. In this case one can choose one arbitrary but fixed element S that has T as a face.*

So Σ_h can be written as the union of all such intersections

$$\Sigma_h = \cup_{T\in\mathcal{F}_h} T, \tag{3.2.2}$$

where $\mathcal{F}_h$ denotes the set of all intersections which can be line segments in 2D, and planar segments in 3D. We also denote with $\mathcal{F}_I$ the set of all internal faces in $\mathcal{K}_h$. Finally, we take as the local mesh size h_S the length of the longest edge in S and as the global mesh size we take $h = \max_{S\in\mathcal{T}_h} h_S$.

Remark 3.2.2. *While we assumed that $\mathcal{T}_h$ consists of elements that are regular, the same is not true for the set $\mathcal{F}_h$ which are not regular in general. The reason is that the elements in $\mathcal{F}_h$ can have very small angles and neighboring elements can have very different lengths or areas. However, this does not influence the optimal error bounds so that regularity in $\mathcal{F}_h$ is not a requirement of the method.*

1 We assume that Σ is a hypersurface without boundary.

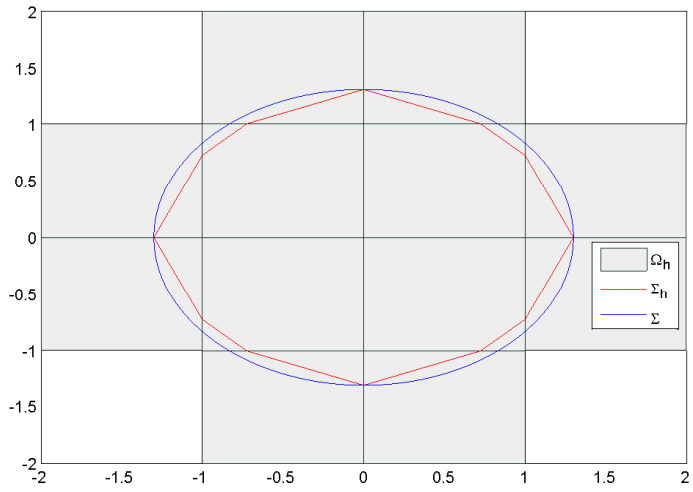

Figure 3: The background mesh Ω_I which consists of squares and contains the circle Σ entirely. The approximated circle Σ_h is obtained using the linear interpolant $\pi_h\phi$ of the implicit function ϕ. The domain Ω_h (shaded squares) contains all intersected elements. The approximated curve Σ_h can be partitioned into line segments $T \in \mathcal{F}_h$.

3.2.1 *Discretization of hypersurface*

An approximation Σ_h of Σ can be obtained by utilizing the implicit representation of the hypersurface. In particular we can represent Σ as the zero level of an implicit function ϕ or a signed distance function ϕ_d as in Section 2.1. The implicit function can then be discretized using piecewise continuous finite elements on the background mesh $\mathcal{T}_h$ so that we obtain ϕ_h. For example we can obtain the discretized implicit function as $\phi_h = \pi_h\phi$ i.e. as the continuous piecewise linear interpolant (A.3.1) of ϕ. Then the discretized curve Σ_h can be obtained as the zero level of ϕ_h. An example discretization of a circle in 2D, using regular quadrilaterals for the partitioning of the background domain, is depicted in Figure 3. For Σ_h we define discretized quantities: the discrete unit normal field (which is the equal to the exact normal on Σ_h)

$$\mathbf{n}_h(\mathbf{x}) = \frac{\nabla\phi_h(\mathbf{x})}{\|\nabla\phi_h(\mathbf{x})\|}, \tag{3.2.3}$$

and the discrete closest point mapping

$$\boldsymbol{\eta}_h(\mathbf{x}) = \mathbf{x} - \phi_{d,h}(\mathbf{x})\mathbf{n}_h(\mathbf{x}), \tag{3.2.4}$$

where $\phi_{d,h}$ is the discretized signed distance function. We can also define approximate differential operators such as the tangential gradient

$$\nabla_{\Sigma_h}\phi(\mathbf{x}) = \mathbf{P}_h\nabla\phi(\mathbf{x}) = (\mathbf{I} - \mathbf{n}_h \otimes \mathbf{n}_h)\nabla\phi(\mathbf{x}). \tag{3.2.5}$$

3.2.2 *Finite element formulation*

The main idea of the cut finite element method is to use the finite element space which is induced by the finite elements on $\mathcal{T}_h$. This surface finite element space is the space of traces on Σ_h of all piecewise linear continuous functions with respect to the background mesh T_h. We introduce the finite element space

$$V_h = \{v_h \in C(\Omega_h) : v_h|_S \in P \ \ \forall\, S \in \Omega_h\}, \tag{3.2.6}$$

where P is the space of polynomials. The space V_h induces the surface space on Σ_h

$$V_h^\Sigma = \{\psi_h \in H^1(\Sigma_h) : \exists\, v_h \in V_h \text{ s.t. } \psi_h = v_h|_{\Sigma_h}\}. \tag{3.2.7}$$

If we denote by ϕ_i, $i = 1, \dots, N$ the nodal finite element basis functions that correspond to the vertices of the elements in Ω_h, then the space V_h^Σ is spanned by the traces of the basis functions $\phi_i|_{\Sigma_h}$. We apply the method to the weak form of the Laplace-Beltrami problem as stated in Section 2.5.1.1 and we obtain the formulation: find $u_h \in V_h^\Sigma$ such that

$$a_h(u_h, v_h) = l_h(v_h) \quad , \quad \forall v_h \in V_h^\Sigma, \tag{3.2.8}$$

where the symmetric bilinear form is

$$a_h(u, v) = (\nabla_{\Sigma_h} u, \nabla_{\Sigma_h} v)_{\Sigma_h}, \tag{3.2.9}$$

and the linear form is

$$l_h(v) = (f^e, v)_{\Sigma_h}, \tag{3.2.10}$$

with f^e being an appropriate extension of f using the closest point mapping (2.1.8), so that $f^e(\mathbf{x}) = f \circ \eta(\mathbf{x}) = f(\eta(\mathbf{x}))$.

Remark 3.2.3. *The extension of functions to a neighborhood U of Σ can be achieved by taking constant values along the normal direction $\mathbf{n}$ to Σ. Note however, that the tangential differential operators are independent of the choice of U so we can extend all functions by taking constant values along the approximate normal $\mathbf{n}_h$ instead.*

Since we assumed that the hypersurface is closed we also add the additional constraint that

$$\int_{\Sigma_h} u_h \, d\mathbf{x}_h = 0. \tag{3.2.11}$$

The solution of (3.2.8) can be expressed as the linear combination of the traces of the outer basis functions on Σ_h

$$u_h = \sum_{i=1}^{N} \xi_i \phi_i|_{\Sigma_h}. \tag{3.2.12}$$

Then we can obtain the linear system $A\xi = b$ with

$$A_{ij} = a_h(\phi_i, \phi_j), \tag{3.2.13}$$

and

$$b_i = l_h(\phi_i). \tag{3.2.14}$$

3.2.3 *Optimal error bounds*

The finite element solution u_h in (3.2.8) obeys the following error bounds[2] [ORG09]:

$$\|u^e - u_h\|_{L^2(\Sigma_h)} \leqslant Ch^2 \|f\|_{L^2(\Sigma)}, \tag{3.2.15}$$

$$\|\nabla_{\Sigma_h}(u^e - u_h)\|_{L^2(\Sigma_h)} \leqslant Ch \|f\|_{L^2(\Sigma)}. \tag{3.2.16}$$

3.2.4 *Stabilization*

The stiffness matrix A can become ill-conditioned depending on the position of the hypersurface relative to the background domain due to the strong shape irregularity of the hypersurface discretization. One solution for this problem is to use a scaled matrix [OR10]. Another solution, which we will follow, is to add a stabilization term. We define this term as

$$j(u, v) = \sum_{F \in \mathcal{F}_I} ([\mathbf{n_F} \cdot \nabla u], [\mathbf{n_F} \cdot \nabla v])_F, \tag{3.2.17}$$

where $\mathbf{n}_F$ is the outward unit normal to the face F and $[\mathbf{n}_F \cdot \nabla u]$ denotes the jump of the gradient along the normal direction[3] when passing an internal face of an element in $\mathcal{K}_h$. If we consider two adjacent elements S^+ and S^- that share a face F, then $\mathbf{n}_F^+$ is the outward unit normal for S^+ and $\mathbf{n}_F^-$ is the outward unit normal for S^-. We have that $\mathbf{n}_F^+ = -\mathbf{n}_F^-$ and the jump of a quantity w is expressed as

$$[\mathbf{n}_F \cdot w] = \mathbf{n}_F^+ \cdot (w^+ - w^-) = \mathbf{n}_F^+ \cdot [w]. \tag{3.2.18}$$

We add the term $j(u_h, v)$ to obtain the stabilized problem: find $u_h \in V_h^\Sigma$ such that

$$a_h(u_h, v_h) + j(u_h, v_h) = l_h(v_h) \quad , \quad \forall v_h \in V_h^\Sigma. \tag{3.2.19}$$

Note that the condition number (A.2.5) of the augmented stiffness matrix $\kappa(A)$ obeys the following bound

$$\kappa(A) \leqslant Ch^{-2}. \tag{3.2.20}$$

3.2.5 *Implementation details*

We describe our implementation of the cut finite element method for the Laplace-Beltrami problem defined on a closed curve which is embedded in 2D space.

2 Given conditions that assure that Σ_h is sufficiently close to Σ.

3 Note that the gradient is continuous along the tangential direction.

CURVE REPRESENTATION For the representation of the curve Σ we use the implicit representation method. In particular we represent the curve as the zero level of a signed distance function ϕ_d as in (2.1.3).

BACKGROUND MESH AND APPROXIMATE CURVE For the background domain Ω_I we use a rectangular domain that completely contains Σ. We then partition this domain into regular rectangles or squares. The approximate curve Σ_h is obtained as the zero level of the approximate signed distance function which is chosen as $\phi_{d,h} = \pi_h \phi_d$. An example construction is depicted in Figure 3.

CONSTRUCTION OF Ω_h We first obtain the intersection points of the curve Σ_h with the background elements using the computation of the zero isocontour. We then construct the set Ω_h, which is the set of all intersected elements, by testing in which element every midpoint of the line segments of Σ_h lies. In our implementation we make the assumption that each T does not coincide with a face in $\mathcal{T}_h$ for simplicity reasons.

BILINEAR FINITE ELEMENTS We choose to use bilinear quadrilateral finite elements. The space of bilinear functions on each element S is

$$\begin{aligned} P(S) = \{v : v = c_0 + c_1 x + c_2 y + c_3 xy\,, \\ (x,y) \in S,\ c_0, c_1, c_2, c_3 \in \mathbb{R}\}, \end{aligned} \tag{3.2.21}$$

and each $v \in P(S)$ is uniquely determined by its nodal values on the four vertices N_i of S. Using the definition of the nodal basis $\phi_j(N_i) = \delta_{ij}$ we obtain the expression for the basis functions of the bilinear quadrilateral element

$$\phi_1 = (x(N_2)y(N_3) - y(N_3)x - x(N_2)y + xy)/E, \tag{3.2.22a}$$

$$\phi_2 = (-x(N_1)y(N_3) + y(N_3)x + x(N_1)y - xy)/E, \tag{3.2.22b}$$

$$\phi_3 = (x(N_1)y(N_1) - y(N_1)x - x(N_1)y + xy)/E, \tag{3.2.22c}$$

$$\phi_4 = (-x(N_2)y(N_1) + y(N_1)x + x(N_2)y - xy)/E, \tag{3.2.22d}$$

where E is the area of the element and $(x(N_i), y(N_i))$ are the (x,y) coordinates of the node N_i in element S.

QUADRATURE For the numerical integration we use Gauss-Legendre n-point quadrature. Since Gaussian quadrature is constructed

to give exact results for polynomials of degree $2n-1$, we choose n accordingly. For example, when we need to calculate the term (3.2.9) which involves gradients, the degree of the polynomial is two, so we choose $n=2$.

ENFORCEMENT OF CONSTRAINT Since we are dealing with a closed curve, we need to take into account the zero mean constraint (3.2.11). We choose to enforce this using the Lagrange multiplier method. In this method we first consider the constraint as a matrix C acting on the solution

$$\int_{\Sigma_h} u_h \, dx_h = \sum_{i=1}^{N} \xi_i \int_{\Sigma_h} \phi_i \, dx_h = C\xi = 0. \tag{3.2.23}$$

We then assemble an augmented linear system as follows

$$\begin{pmatrix} A & C^T \\ C & 0 \end{pmatrix} \begin{pmatrix} \xi \\ \mu \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}, \tag{3.2.24}$$

where A is the stiffness matrix, μ is the Lagrange multiplier, b is the load vector. Therefore we seek a solution $[\xi, \mu]$ of the augmented linear system. Note that the Lagrange multiplier μ should be zero or very small because the constraint does not alter the solution to the initial problem.

EIGENVALUES AND CONDITION NUMBER We calculate the effective condition number of the stiffness matrix using the ratio of the largest and first positive eigenvalues

$$\kappa_{\text{eff}}(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min>0}(A)}. \tag{3.2.25}$$

The first eigenvalue is negative as it corresponds to the Lagrange multiplier. The first non-zero eigenvalue is the second one in the stabilized method and the third one in the unstabilized method. The second eigenvalue in the unstabilized method is zero because we use the same background mesh to define the approximate curve and to perform the computations. This can be avoided if we instead use a finer mesh to define the approximate curve.

IMPLEMENTATION STRUCTURE The implementation structure follows the classic finite element method structure i.e. we exploit the limited support of the basis functions and assemble the global matrices by adding local contributions from each element. To this purpose we use connectivity arrays which map local to global degrees of freedom. The steps of our implementation are the following

1. Find the intersection points and the intersected elements.
2. Find the tangent and normal vectors to each line segment of Σ_h.
3. Assemble the stiffness matrix, the load vector, the constraint term matrix and the stabilization term matrix.
4. Solve the augmented Lagrange multiplier linear system, visualize the solution and calculate the error norms.

PERFORMANCE Vectorization [TLNC10] is used wherever possible to improve performance. However, parts of the code are not optimal in the sense that the algorithms used are not the most efficient. This is for example the case for the intersection routine which calculates which elements are intersected by the curve. A more efficient implementation of this should involve the use of quadtree or octree data representations. More information on this subject can be found in Massing, Larson and Logg [MLL12].

Example listings of the MATLAB code can be found in Appendix B.

4

ISOGEOMETRIC ANALYSIS FOR PDES ON MANIFOLDS

4.1 INTRODUCTION TO ISOGEOMETRIC ANALYSIS

Isogeometric analysis is a computational method that is based on the isoparametric paradigm so that it uses the same basis functions for the geometric representation of an object and the solution space in the numerical analysis. The initial aim was to use CAD models directly without the need to generate approximate geometrical descriptions. Due to the wide-spread use and the effectiveness of splines in CAD analysis, isogeometric analysis is mainly based on B-splines and non-uniform rational B-splines (NURBS). Using NURBS many geometric shapes such as conics can be parametrically represented exactly and thus the difficulty and the error produced from the need to generate an approximate description are removed.

Isogeometric analysis has additional advantages over classical finite element methods with one of the most important being the availability of arbitrary degrees of inter-element continuity, in contrast with the usual C^0 continuity in FEM. Another advantage is that the geometrical description is fixed at coarsest level of the discretization i.e. it stays the same throughout refinement. The idea is pictured in Figure 4. Isogeometric analysis can be thought of as a superset of classical FEM as it provides additional tools and improved accuracy and efficiency. For example, in addition to *h*-refinement (knot insertion) and *p*-refinement (order elevation) which have an analogue in FEM, there is also an additional refinement process called *k*-refinement. *K*- refinement

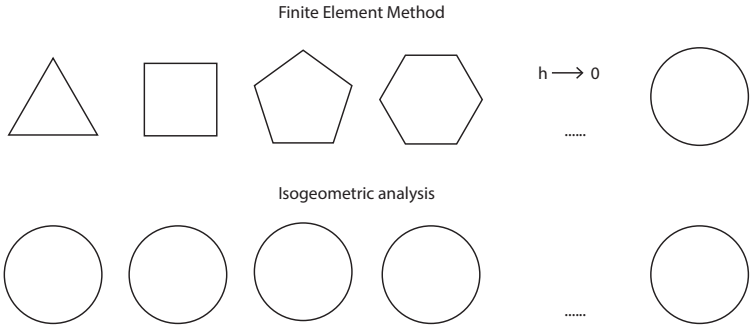

Figure 4: In FEM the geometrical description of an object approaches the desired form as $h \to 0$. In isogeometric analysis it is exact and fixed at the coarsest level of discretization.

is based on the fact that the two aforementioned refinement processes do not commute and it doesn't have an analogue in FEM.

While the isoparametric concept is also used in classical finite element analysis, there is an important difference. In FEM the basis chosen for the solution space is also chosen to represent the geometry. In IGA the basis chosen to represent the geometry is also chosen to represent the solution. Additionally, in FEM each element in the physical space has its own mapping from the reference element. This is in contrast to IGA, where a single mapping takes an entire patch[1] from the parameter space to the physical space.

There are many excellent references on the subject of NURBS [Rog01, PT97] and IGA [HCB09, NSBR12, VHS10]. Here we will cover the basic definitions that are essential to our purpose.

4.1.1 *Basic isogeometric analysis concepts*

Definition 4.1.1. *A knot vector* $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$ *in one dimension is a non-decreasing set of coordinates in the parameter space, where* $\xi_i \in \mathbb{R}$ *is the* i^{th} *knot,* p *is the polynomial degree*[2] *and* n *is the number of basis functions used to construct the B-spline curve.*

We usually assume that $\xi_1 = 0$ and $\xi_{n+p+1} = 1$. The knot vector partitions the parameter space into "elements" called knot spans written as $[\xi_i, \xi_{i+1})$. A knot ξ_i can be repeated m_i times, in which case we refer to the *multiplicity* m_i of this knot. A knot vector is said to be open if the first and last knots have multiplicity equal to $(p+1)$. A knot vector is called uniform when the knots are evenly distributed, otherwise it is called non-uniform.

Definition 4.1.2. *Given a knot vector* Ξ*, the B-spline basis functions are defined recursively starting with piecewise constants*

$$N_{i,0}(\xi) = \begin{cases} 1 & \text{if } \xi_i \leqslant \xi < \xi_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{4.1.1}$$

and for polynomial order $p \geqslant 1$

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi). \tag{4.1.2}$$

This is the Cox-de Boor recursion formula. Note that by convention we take that ratios of $0/0$ *are equal to zero.*

1 The patch can be thought as a sub-domain which in the parameter space takes the form of a rectangle in 2D or a cuboid in 3D. In most cases, a single patch is sufficient to model the geometry and is comprised of elements called *knot spans*.

2 We adhere to the convention that order equals degree.

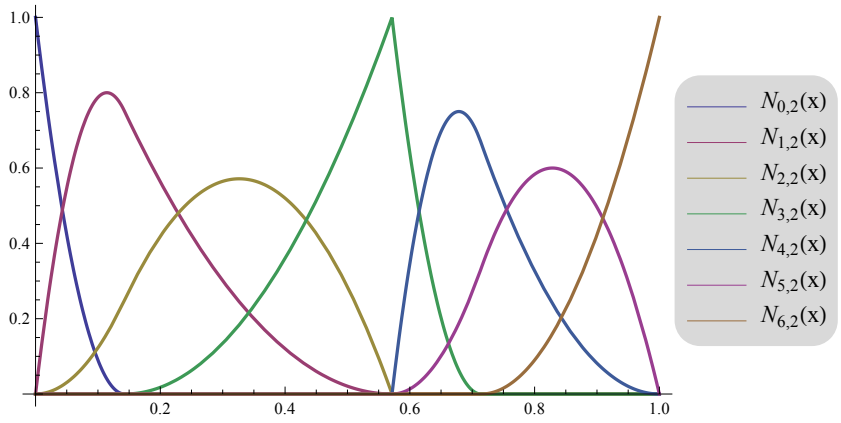

Figure 5: Example univariate B-spline basis functions for the open knot vector $\Xi = \{0, 0, 0, 1/7, 4/7, 4/7, 5/7, 1, 1, 1\}$. The order is $p = 2$ and there are $n = 7$ basis functions. We can see that the basis functions are interpolatory at the ends and at the knot $\xi = 4/7$ where the multiplicity is $m = 2$, so that the continuity there is $C^{p-m} = C^{2-2} = C^0$.

B-splines have many important properties

- The B-spline basis constitutes a partition of unity i.e. $\sum_{i=1}^{n} N_{i,p}(\xi) = 1$.
- They are non-negative over the entire domain.
- Basis functions of order p have $p - m_i$ continuous derivatives across the knot ξ_i, where m_i is the multiplicity of the knot ξ_i.
- Basis functions are generally non-interpolatory. Only when $m_i = p$, the knot ξ_i is interpolatory. This is always true (for an open knot vector) at the endpoints where the basis becomes discontinuous and it creates the patch boundary.
- The support of the basis functions of order p is always $(p + 1)$ knot spans. At any given knot span $[\xi_i, \xi_{i+1})$, the non-zero functions are $N_{i-p,p}, \ldots, N_{i,p}$.

Definition 4.1.3. *Given n B-splines basis functions $N_{i,p}$ and control points $\mathbf{B}_i \in \mathbb{R}^n$, a piecewise-polynomial B-spline curve is given by*

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi)\mathbf{B}_i. \tag{4.1.3}$$

Having defined univariate B-spline basis functions and B-splines curves, we can now use the tensor product to construct multivariate B-spline basis and B-spline surfaces.

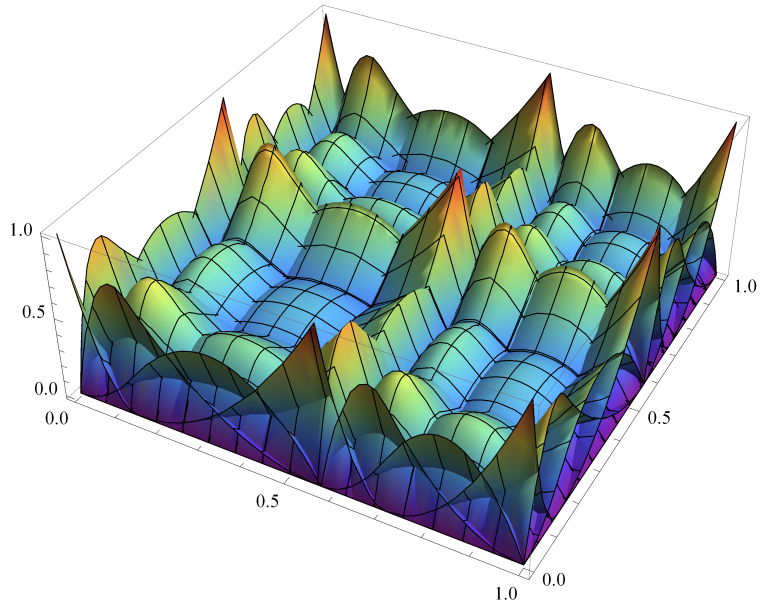

Figure 6: Example multivariate B-spline basis functions for the tensor product $\Xi \times \Xi$ with $\Xi = \{0, 0, 0, 1/7, 4/7, 4/7, 5/7, 1, 1, 1\}$. The order is $p = 2$ and there are $n = 49$ basis functions.

Definition 4.1.4. *Given two knot vectors* $\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}$ *and* $H = \{\eta_1, \eta_2, \ldots, \eta_{m+q+1}\}$ *and a control net* $\mathbf{B}_{i,j} \in \mathbb{R}^n$*, a tensor-product B-spline surface is defined as*

$$\mathbf{S}(\xi, \eta) = \sum_{i=1}^{n} \sum_{j=1}^{m} N_{i,p}(\xi) M_{j,q}(\eta) \mathbf{B}_{i,j}, \tag{4.1.4}$$

where $N_{i,p}(\xi)$ *and* $M_{j,q}(\eta)$ *are the univariate B-spline basis functions of order* p *and* q *corresponding to knot vectors* Ξ *and* H*, respectively.*

We now define NURBS, which are a generalization of B-splines and are able to exactly represent simple shapes such as conics.

Definition 4.1.5. *NURBS basis functions are defined by*

$$R_{i,p}(\xi) = \frac{N_{i,p}(\xi) w_i}{W(\xi)} = \frac{N_{i,p}(\xi) w_i}{\sum_{j=1}^{n} N_{j,p}(\xi) w_j}, \tag{4.1.5}$$

where w_i *is referred to as the* i^{th} *weight and is always positive.*

NURBS curves and surfaces are piecewise rational functions and are defined similarly as B-spline curves and surfaces.

Remark 4.1.1. *NURBS entities in* $\mathbb{R}^n$ *are obtained by projective transformations of B-spline entities in* $\mathbb{R}^{n+1}$*.*

As we mentioned, there are three types of refinement in isogeometric analysis:

H-REFINEMENT This is also referred to as knot insertion as new knots are inserted without changing the geometry. The result is a richer basis (more elements, basis functions and control points) for the solution space.

P-REFINEMENT This is achieved by order elevation of the basis functions. To maintain discontinuities, the multiplicity of each knot is also increased by one. Again this does not change the geometry but only the solution space.

K-REFINEMENT This is achieved by first elevating the order of the basis functions and then inserting new knots. The result is increased continuity and polynomial order of the basis functions. This process has no direct analogue in classical FEM.

A disadvantage of IGA is that due to tensor product nature of the basis functions, refinement is a global procedure. This leads to superfluous control points. Possible solutions are the use of T-splines [BCC+10] and hierarchical refinement [VGJS11].

4.2 ISOGEOMETRIC ANALYSIS FOR PDES ON MANIFOLDS

4.2.1 *Hypersurface representation in isogeometric analysis*

As we described above, IGA uses B-splines or NURBS to parametrically represent the geometry as in (2.2.1). The parameter space $\widehat{\Sigma}$ is defined by the knot vector Ξ for a curve or by the tensor product of two knot vectors $\Xi \times H$ for a surface. The knot vectors define the elements (unique knots spans) $\widehat{\Sigma}_e$, $e = 1, \dots, n_{el}$ in the parameter space and through the geometrical mapping, the elements Σ_e in the physical space. We will also define the mesh size h_e as the maximal diameter of element Σ_e and the global mesh size as $h = \max(h_e, e = 1, \dots, n_{el})$. We can write the geometrical mapping as

$$\mathbf{x}(z) = \sum_{i=1}^{n} \widehat{R}_i(z)\mathbf{B}_i, \tag{4.2.1}$$

where $\widehat{R}_i(\xi)$ are the NURBS basis functions defined in the parameter space $\widehat{\Sigma}$ and $\mathbf{B}_i \in \mathbb{R}^n$ are the control points. Here z in general denotes a multivariate knot vector $z = (z^1, \dots, z^\kappa)$ where κ is the dimension of the parameter space. Similarly, $\widehat{R}_i(z)$ are in general multivariate basis functions. The geometrical mapping is depicted in Figure 7.

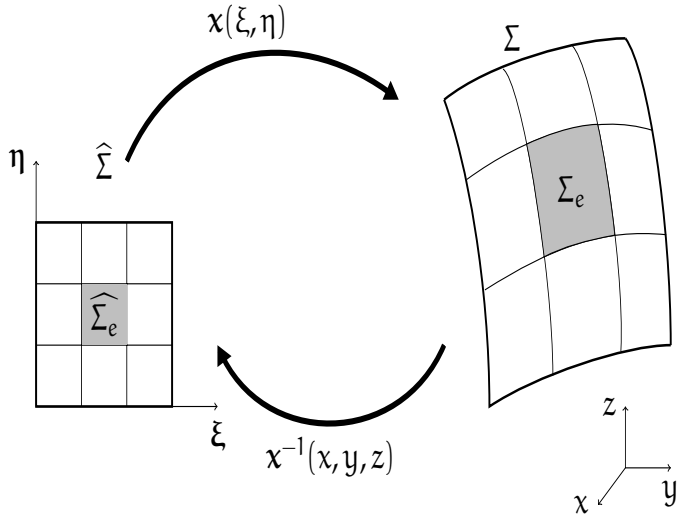

Figure 7: Geometrical mapping in Isogeometric Analysis: The surface in 3D space is constructed using a mapping $\mathbf{x}$ from the parameter space. An element $\widehat{\Sigma}_e$ in parameter space is mapped to an element Σ_e in physical space.

4.2.2 *Galerkin method for isogeometric analysis*

IGA also uses the same basis functions for the solution space. So we can write the approximate solution u_h as

$$u_h(\mathbf{x}) = \widehat{u}_h(\mathbf{x}) \circ \mathbf{x}^{-1}(\xi) = \sum_{i=1}^{n} \widehat{R}_i(z) d_i, \tag{4.2.2}$$

where $d_i \in \mathbb{R}$ are the unknown coefficients which are called control variables. The Galerkin method take the same form as in Section 3.1 but using NURBS basis functions for the solution space. The finite element space $\widehat{V}_h$, defined on the parametric domain, is spanned by the NURBS basis functions

$$\widehat{V}_h = \operatorname{span}\{\widehat{R}_i\}_{i=1}^{n}. \tag{4.2.3}$$

The finite element space V_h, defined on the physical domain, can be deduced by $\widehat{V}_h$ as

$$V_h = \operatorname{span}\{\widehat{R}_i \circ \mathbf{x}^{-1}\}_{i=1}^{n}. \tag{4.2.4}$$

4.2.3 *A priori error estimates*

A priori estimates for isogeometric analysis have been derived [BBdVC+06, dVBRS11] and the results state that the solution u_h satisfies the same optimal rate of convergence as the classical finite element method of degree p. In general, we assume that these optimal convergence rates are also obeyed for the isogeometric analysis of PDEs on manifolds, as long as the geometrical

mapping $\mathbf{x}$ is sufficiently smooth. Specific error estimates for the Laplace-Beltrami problem were derived in Dedè and Quarterioni [DQ13]:

$$\|u - u_h\|_{L^2(\Sigma)} \leqslant Ch^{p+1}, \tag{4.2.5}$$

$$\|u - u_h\|_{H^1(\Sigma)} \leqslant Ch^{p}, \tag{4.2.6}$$

where p is the polynomial order of the NURBS basis functions. For the Laplace-Beltrami eigenvalue problem, an error estimate for the n^{th} eigenvalue is

$$\lambda_{n,h} - \lambda_n \sim C(\lambda_n)^{p+1} h^{2p}. \tag{4.2.7}$$

For higher order problems such as the biharmonic problem, we assume the error estimates follow the ones provided in Tagliabue et al. [TDQ13]. These state that the following error estimates hold:

$$\|u - u_h\|_{L^2(\Sigma)} \leqslant Ch^{\min\{p+1,2p-2\}}, \tag{4.2.8}$$

$$\|u - u_h\|_{H^1(\Sigma)} \leqslant Ch^{\min\{p,2p-2\}}, \tag{4.2.9}$$

$$\|u - u_h\|_{H^2(\Sigma)} \leqslant Ch^{\min\{p-1,2p-2\}}. \tag{4.2.10}$$

Finally, the condition number of the stiffness matrix in IGA obeys the error bound [GT12]

$$\kappa(A) \leqslant Ch^{-2}. \tag{4.2.11}$$

4.2.4 *Implementation details*

As mentioned, the implementation of the isogeometric analysis method was done in MATLAB by extending the package GeoPDEs. The main additional feature that was needed for the implementation, was the support for cases where the dimension κ of the parameter space and the dimension n of the physical space in the mapping (2.2.1) are not equal. This has consequences for the numerical calculation of all quantities as the determinant of the Jacobian has to be replaced with the square root determinant of the first fundamental form. When $\kappa = n$ these two quantities are equal, so we can say that the case $\kappa \neq n$ is a generalization.

The implementation of IGA is similar as in isoparametric FEM. The main difference lies in the construction of the connectivity arrays and in the evaluation of the shape functions. An alternative approach is to use Bézier extraction, which produces Bézier elements [SBV+11]. These can be used in the same way as in standard finite element method implementations.

QUADRATURE The numerical calculation of integrals in IGA is performed by pulling back quantities onto the parameter space and then using a quadrature rule defined on a reference element. For example in the assembly of the stiffness matrix, we need to be able to calculate integrals of the form

$$\alpha(u, v) = \int_{\Sigma} \nabla_{\Sigma} u \cdot \nabla_{\Sigma} v \, dx. \tag{4.2.12}$$

We first pull-back the integral onto the parameter domain using the equations in Section 2.4.1 so it becomes

$$\widehat{\alpha}(\widehat{u}, \widehat{v}) = \int_{\widehat{\Sigma}} \widehat{\nabla} \widehat{u} \cdot (\widehat{G}^{-1} \widehat{\nabla} \widehat{v}) \widehat{g} \, d\xi. \tag{4.2.13}$$

Similarly we convert $l(v)$ to

$$\widehat{l}(\widehat{v}) = \int_{\widehat{\Sigma}} \widehat{f} \, \widehat{v} \, d\xi. \tag{4.2.14}$$

The integrals are then calculated using Gauss-Legendre n-point quadrature which is initially defined on a reference element and then transformed to the parameter space. We choose $n = (p+1)^{\kappa}$ abscissas where p is the polynomial order of the NURBS basis functions and κ is the dimension of the parameter space.

Remark 4.2.1. *Although the choice of $n = (p+1)^{\kappa}$ quadrature points guarantees exact integration, it is not optimal complexity-wise as the increased continuity between elements in IGA can be exploited to produce a quadrature rule which spans more than one element, so that it requires less evaluation points per degree of freedom [HRS10].*

BOUNDARY CONDITIONS AND CONSTRAINT For the enforcement of the zero mean constraint for closed manifolds, we use the same Lagrange multiplier method as in FEM, namely (3.2.24). The imposition of homogeneous Dirichlet boundary conditions in open manifolds is similar as in classical FEM. We find the degrees of freedom associated to the boundary and set them identically equal to zero. Then we solve the linear system for the internal degrees of freedom. For the biharmonic problem the situation is more complex as we need to enforce the two conditions (2.5.8b) and (2.5.8c). Since the NURBS basis functions are interpolatory at their ends, we can enforce the condition (2.5.8c) by setting identical values for the boundary and boundary-adjacent degrees of freedom. In particular, we set these to zero and solve for the remaining internal degrees of freedom.

Remark 4.2.2. *For closed manifolds there are coinciding control points. For example, the first and last control points in the circle coincide. This is due to the use of open aperiodic knot vectors. Therefore we need to enforce that the degrees of freedom corresponding to coinciding control points, coincide as well. This can be achieved by identifying such points and manipulating the connectivity array. Moreover, in the solution of the linear system, we identify such "duplicated" degrees of freedom and ignore them.*

TIME DISCRETIZATION For the time-dependent advection diffusion problem we discretize time by dividing into time steps $t_i, i = 0, \dots, n$ with $t_0 = 0$ and $t_n = T$ so that the time step Δt is constant. We then use the Crank-Nicholson method for the time stepping of the solution. The linear system for the unstabilized problem becomes

$$M\frac{\xi_i - \xi_{i-1}}{\Delta t} + A\frac{\xi_i + \xi_{i+1}}{2} + C\frac{\xi_i + \xi_{i+1}}{2} = \frac{b_i + b_{i+1}}{2}, \tag{4.2.15}$$

for each time step i, where M is the mass matrix corresponding to the term $m(u, v)$, A is the stiffness matrix corresponding to the term $\alpha(u, v)$, C is the matrix corresponding to the advection term $\beta(u, v)$ and b is the load vector corresponding to the term $l(v)$. Note that in the first time step we choose that u_0 is the zero vector.

SUPG STABILIZATION For the time-dependent advection-diffusion problem (2.5.12) we implement SUPG stabilization by adding the following term on the left hand side of (3.1.1)

$$\mathcal{L}_h(u_h, v_h) = \sum_{e=1}^{nel} \tau_e(-\nabla_\Sigma \cdot (\mu \nabla_\Sigma u_h) + \mathbf{b} \cdot \nabla_\Sigma u_h - f, \mathbf{b} \cdot \nabla_\Sigma v_h)_{L^2(\Sigma_e)} \tag{4.2.16}$$

where the stabilization parameter τ_e is chosen as

$$\tau_e = c\left(\frac{1}{\Delta t^2} + \left(\frac{V_e}{h_e}\right)^2 \left(1 + \left(\frac{c_p}{\mathcal{P}_{e_e}}\right)^2\right)\right)^{-1/2}, \tag{4.2.17}$$

with c being a constant, Δt being the time step, h_e the element mesh size, $V_e = \|\mathbf{b}\|_{L^\infty(\Sigma_e)}$ and c_p a constant that depends[3] on the order p of the basis functions. Also $\mathcal{P}_{e_e} = \frac{V_e h_e}{2\mu}$ is the local Péclet number which measures the relative strength of the diffusion and advection terms. When advection dominates we have that $\mathcal{P}_{e_e} \gg 1$.

3 We choose $c_p = p^2$.

IMPLEMENTATION STRUCTURE GeoPDEs is based on object oriented programming and as such there are three important classes

- *The geometry class*. This defines the geometry of the physical domain by taking as input a NURBS structure and by defining methods to compute the geometrical mapping and its derivatives.
- *The mesh class*. This defines the parameter space partitioning into elements and calculates the quadrature points and weights on each element using a quadrature rule.
- *The space class*. This contains the information regarding the basis functions of the finite element space V_h. Since we base our analysis on the isoparametric paradigm, the information for V_h is already available from the geometry class. Moreover, a connectivity array is utilized to map locally supported basis functions on each element to the global numbering.

Using these classes, we can assemble the required matrices through the use of the corresponding operators. These operators exploit the tensor product nature of the domain partitioning by traversing through columns in one parametric direction and assembling the corresponding matrices . GeoPDEs also includes functions for refinement, calculation of error norms and visualization of solutions.

The work in this thesis includes the following contributions to the extension of the library:

1. Extension of all structures and functions so that the case of $\kappa \neq n$ is supported. This includes support for 1D parametric spaces which were not available in GeoPDEs. It also includes the calculation of the first fundamental form, its inverse and its square root determinant.
2. Generalization of the calculation of the pull-backs onto the parameter space.
3. Implementation of new operators for the solution of the biharmonic problem, the advection-diffusion problem and for SUPG stabilization.
4. Implementation of a function for the calculation of the error in the H^2 norm.

PERFORMANCE The MATLAB code in GeoPDEs and in the implemented extensions is vectorized when possible to improve performance. Additionally, the tensor product nature of the parameter space is exploited and quantities are

calculated only when needed, resulting in significant memory savings. Finally, demanding operations that are used many times are implemented in C using MATLAB MEX-files. The result is optimized code which performs reasonably well for simple problems.

Example listings can be found in Appendix B.

5

NUMERICAL EXAMPLES

5.1 LAPLACE-BELTRAMI PROBLEM FOR A CURVE IN 2D

We consider the Laplace-Beltrami problem as in Equation 2.4.5, defined on a circle which is embedded in 2D space. We choose that the circle is centered at $(x_0, y_0) = (0, 0)$ and has radius $r = 1$. The exact analytical solution u is chosen to be

$$u(r, \phi) = 12 \sin(3\phi), \tag{5.1.1}$$

in polar coordinates (r, ϕ). Plugging this in (2.5.1a), we obtain the source function f

$$f(r, \phi) = 108 \sin(3\phi). \tag{5.1.2}$$

We solve the problem using both the cut finite element method and isogeometric analysis.

5.1.1 *Solution with the cut finite element method*

We follow the solution method as described in Section 3.2. We represent the circle with the signed distance function

$$\phi_d(x, y) = \sqrt{(x - x_0)^2 + (y - y_0)^2} - r, \tag{5.1.3}$$

and embed it in a background rectangular domain $[-2, -2] \times [2, 2]$ which we partition into squares of equal size h as in Figure 8. An example solution is shown in Figure 9. We then move diagonally the position of the circle relative to the background mesh in small increments and calculate the condition numbers for the unstabilized and stabilized numerical solutions. The results, as depicted in Figure 10, clearly show a large variation of the condition number for the unstabilized version depending on the position of the curve relative to background mesh. We also plot the mesh dependence of the condition number in the stabilized version and we confirm the estimate (3.2.20). Finally we perform convergence analysis in the L^2 norm and compare the theoretical order of convergence $\mathcal{O}(h^2)$ derived from the a priori error estimate in (3.2.15). The results are shown in Figure 12 and confirm the optimal order of convergence for both the stabilized method and unstabilized methods (with the stabilized method performing slightly worse than the unstabilized one).

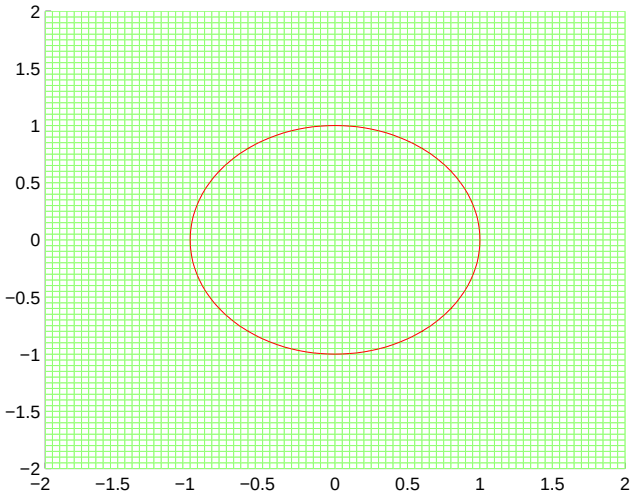

Figure 8: Background mesh of mesh size $h = 0.05$ and approximated unit circle Σ_h. The set Ω_h has nel $= 156$ elements and there are ndof $= 312$ degrees of freedom.

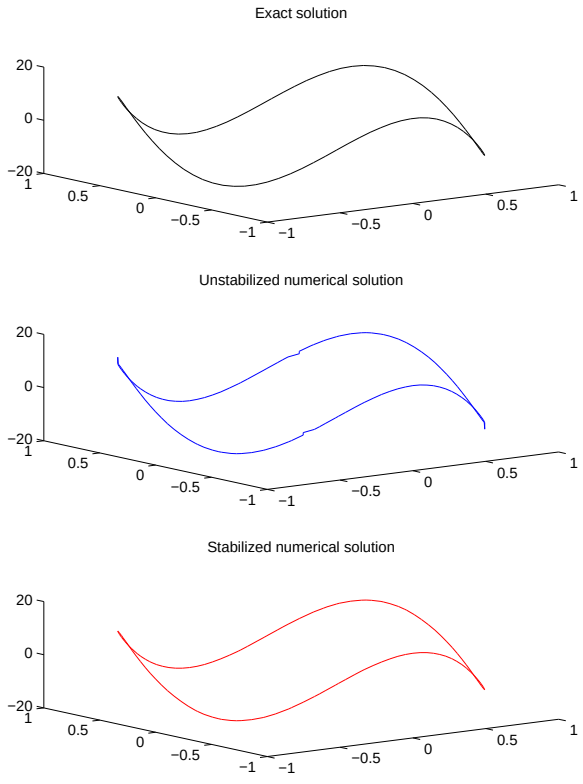

Figure 9: Exact, unstabilized numerical solution and stabilized numerical solution of the Laplace-Beltrami problem for a unit circle in 2D, using the cut finite element method. Note the irregularities in the unstabilized solution.

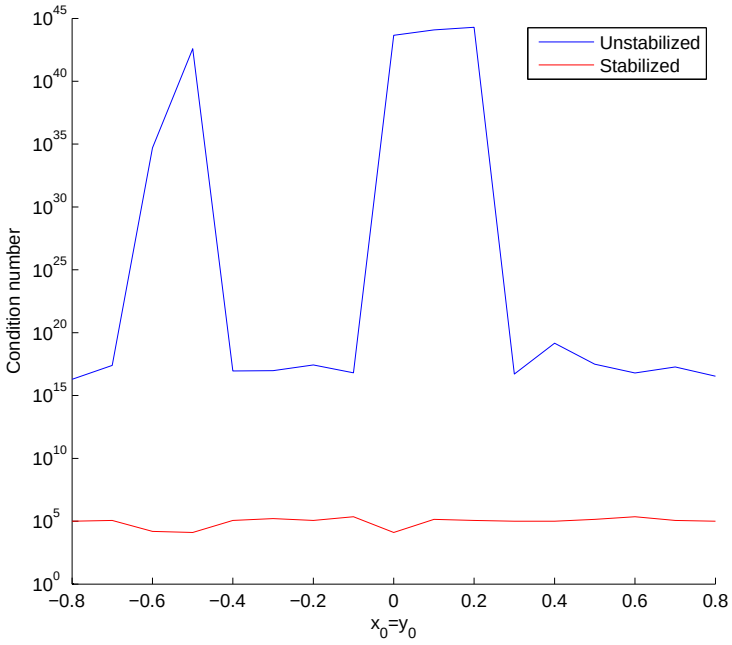

Figure 10: Condition numbers for the stabilized and unstabilized methods as the center of the circle (x_0, y_0) changes position relative to the background mesh. Note the high variation in the unstabilized method.

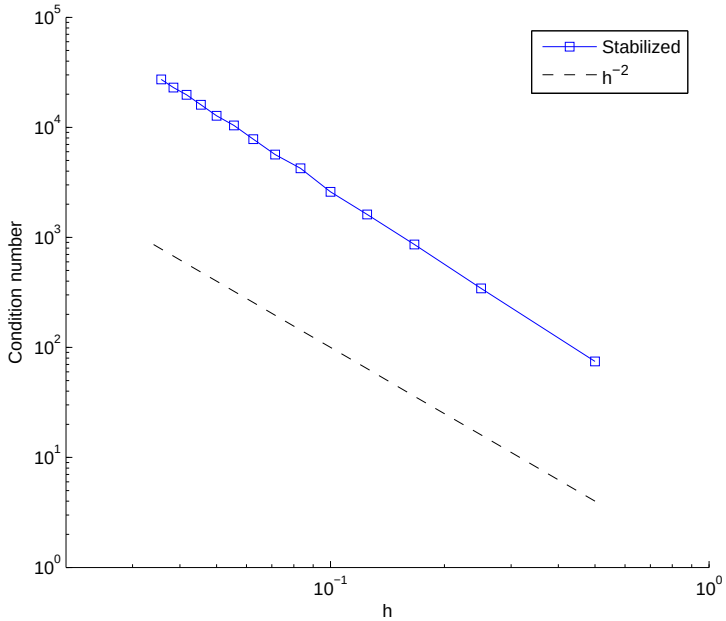

Figure 11: Plot of the condition number $\kappa(A)$ vs. the mesh size h in the stabilized method. The result confirms the theoretical estimate (3.2.20).

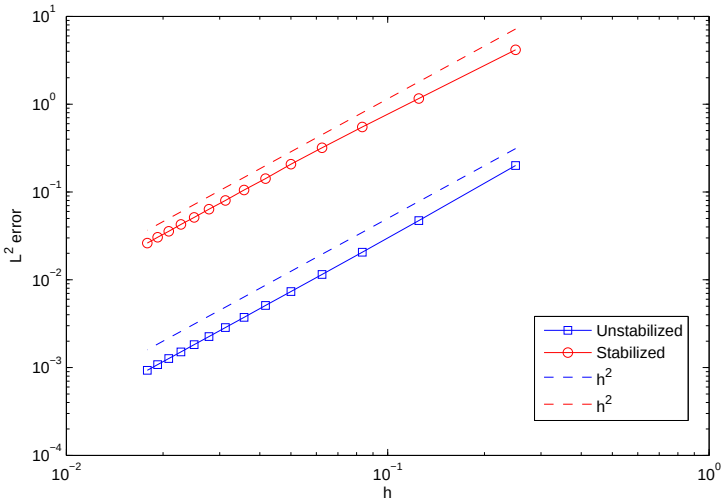

Figure 12: Plot of the L^2 error norm vs. the mesh size h for the Laplace-Beltrami problem using the cut finite element method and comparison with the theoretical convergence rate. Optimal order of convergence $\mathcal{O}(h^2)$ is confirmed for both the stabilized and unstabilized methods.

5.1.2 *Solution with isogeometric analysis*

We solve the same Laplace-Beltrami problem on a curve in 2D using isogeometric analysis with NURBS basis functions of polynomial order $p = 2$ and global C^0 continuity. The geometry, the control points and the elements in the unrefined mesh are shown in Figure 13. An example solution is shown in Figure 14. We then perform convergence analysis in the L^2 and H^1 norms using h-refinement for basis functions of polynomial order $p = 2$. The results shown in Figure 15 confirm the optimal error estimates $\mathcal{O}(h^2)$ for the H^1 norm and $\mathcal{O}(h^3)$ for the L^2 norm as in (4.2.5) and (4.2.6). Finally, we check the mesh dependence of the condition number of the stiffness matrix. The result, depicted in Figure 16, confirms the estimate in (4.2.11).

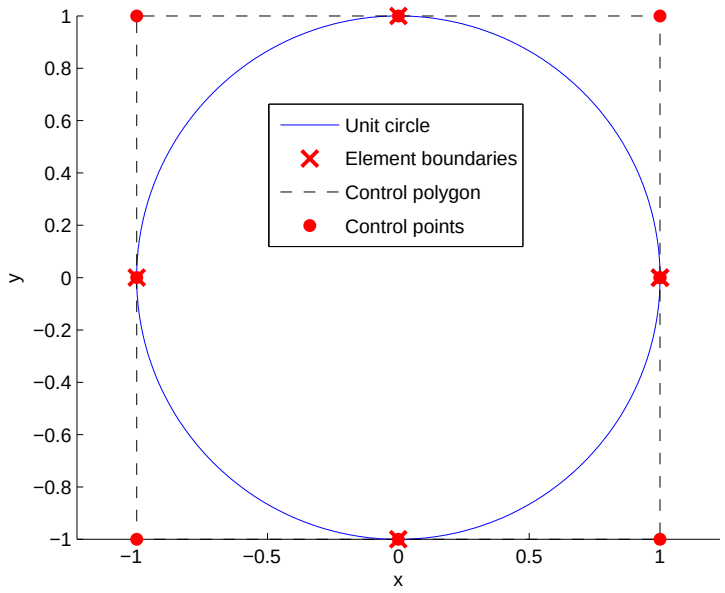

Figure 13: NURBS circle in 2D: exact representation of the unit circle, control points, control polygon and mesh element boundaries in the coarsest level of discretization. There are four elements and nine control points. The first and last control points coincide.

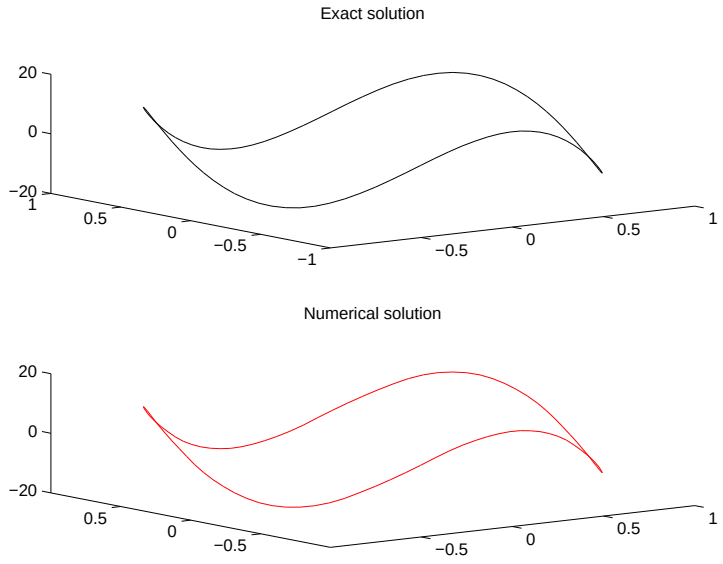

Figure 14: Exact and numerical solution of the Laplace-Beltrami problem for a unit circle in 2D, using isogeometric analysis with NURBS basis functions of order $p = 2$. The number of elements in the refined mesh is $nel = 16$ with number of degrees of freedom $ndof = 312$.

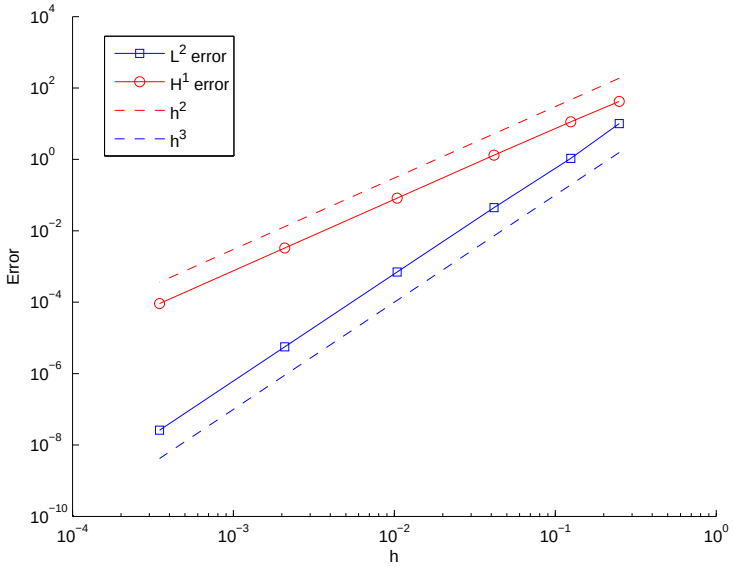

Figure 15: Plot of the L^2 and H^1 error norms for different mesh sizes h, for the Laplace-Beltrami problem in 2D using IGA. The optimal rates of convergence are confirmed.

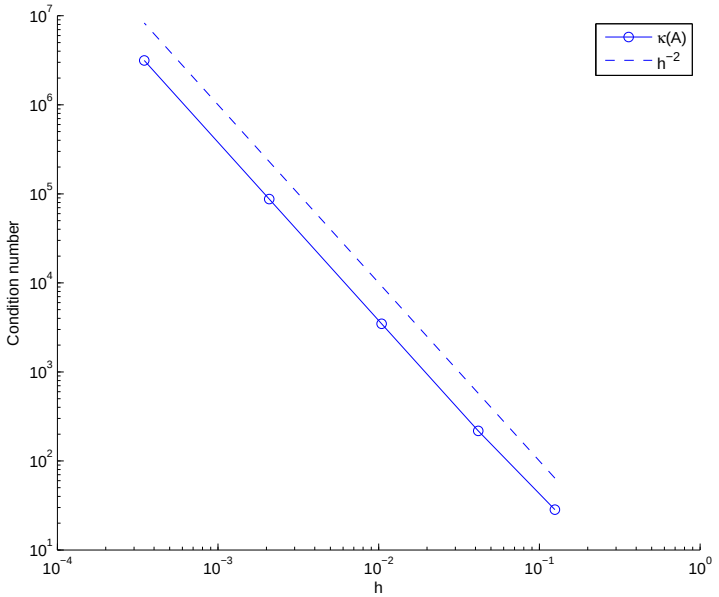

Figure 16: Plot of the condition number $\kappa(A)$ vs. the mesh size h for the Laplace-Beltrami problem in 2D with IGA. The result confirms the theoretical estimate (4.2.11).

5.1.3 *Comparison of IGA and FEM for the Laplace-Beltrami problem in 2D*

We now compare the stabilized cut finite element method and the IGA method with NURBS basis functions of order $p = 2$ for the solution of the Laplace-Beltrami problem on the unit circle in 2D. We report in Table 1 the number of elements, the number of degrees of freedom and the L^2 error for different mesh sizes. We see that the cut finite element method needs a much larger number of degrees of freedom to achieve the same L^2 errors as IGA. The exact geometrical representation and the smoothness of the basis functions in IGA are important advantages in this comparison, as better accuracy is achieved with a smaller number of degrees of freedom.

IGA			FEM		
nel	ndof	L^2 err	nel	ndof	L^2 err
4	9	10.0192	-	-	-
8	17	1.0664	12	24	10.4922
24	49	0.0443	28	56	4.1608
96	193	$7.0311 \cdot 10^{-4}$	92	184	0.5502
480	961	$5.6306 \cdot 10^{-6}$	380	776	0.0054
2880	5761	$2.6069 \cdot 10^{-8}$	-	-	-

Table 1: Comparison of the cut FEM and IGA for the Laplace-Beltrami problem on a circle in 2D. The number of elements, number of degrees of freedom and the L^2 error norm are reported. IGA performs better than the cut finite element method for this type of problem.

5.2 LAPLACE-BELTRAMI EIGENVALUE PROBLEM FOR A CURVE IN 2D

We solve the Laplace-Beltrami eigenvalue problem as in (2.5.2) for the unit circle in 2D with NURBS basis functions of order $p = 2$. The exact eigenvalues for this problem are $\lambda = n^2, n = 1, \ldots, \infty$ each with multiplicity equal to two [Shu01]. The first eigenvalue is zero because the manifold we consider is closed. The exact eigenfunctions can generally be expressed as $u_n = A\cos(n\phi) + B\sin(n\phi)$ where A, B are constants and ϕ is the polar coordinate. We plot the 6^{th} eigenfunction and the ratios of the numerical to exact eigenvalues. The results are shown in Figure 17. The numerical eigenvalues agree very well with the exact eigenvalues for small values of n as one expects from the error estimate in (4.2.7). We also plot in Figure 18 the dependence of

the error on the mesh size h for the fifth eigenvalue and confirm that it follows the estimate for $p = 2$ i.e. that $\lambda_{5,h} - \lambda_5 \sim (\lambda_5)^3 h^4$.

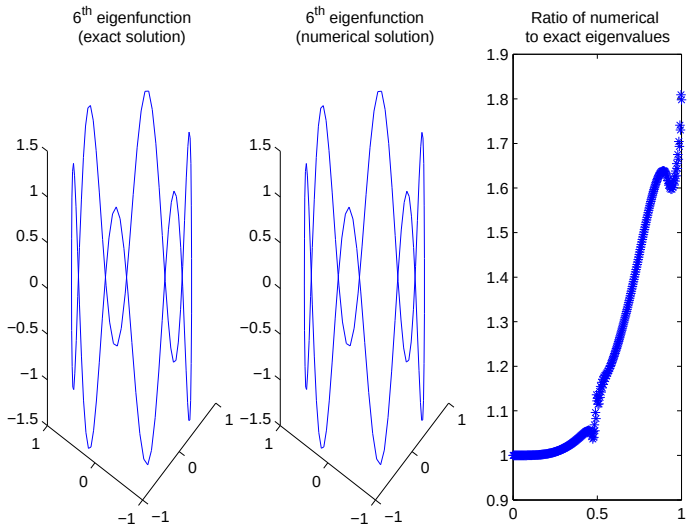

Figure 17: Plot of the 6^{th} eigenfunction and the ratios of the numerical to exact eigenvalues for the Laplace-Beltrami eigenvalue problem on a circle in 2D. For small eigenvalues the numerical results agree with the theoretical values.

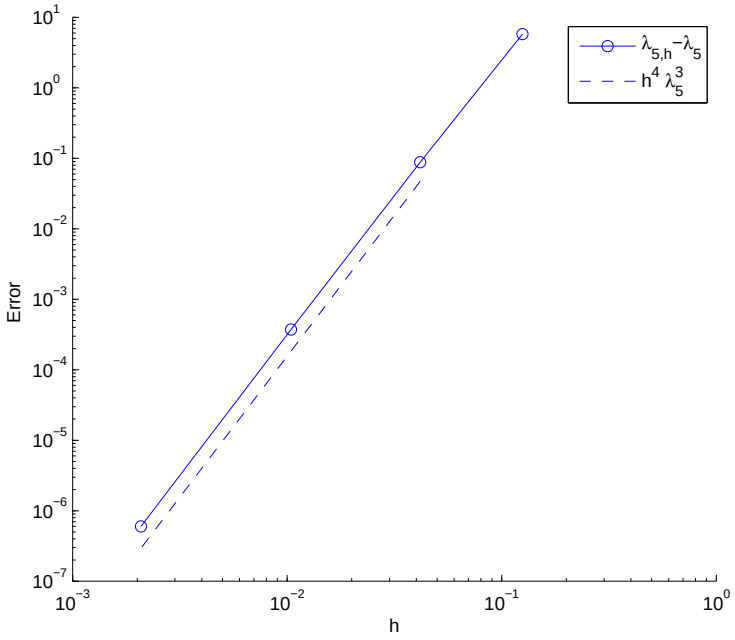

Figure 18: Error vs. mesh size for the fifth eigenvalue of the Laplace-Beltrami eigenvalue problem. NURBS basis functions of order $p = 2$ are used. The theoretical error estimate is confirmed.

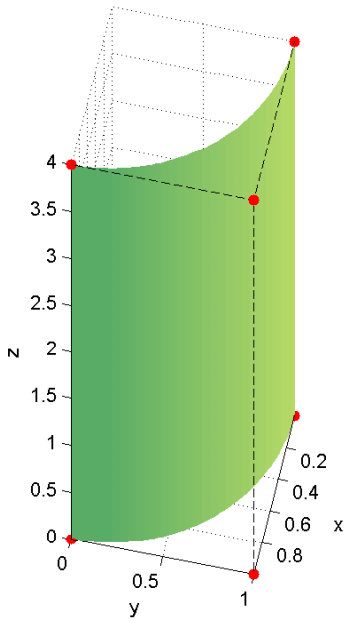

Figure 19: NURBS geometry of quarter cylinder: exact representation, control points and control net in the coarsest level of discretization. There is one element and six control points.

5.3 LAPLACE-BELTRAMI PROBLEM FOR A SURFACE IN 3D

We now solve the Laplace-Beltrami problem for a quarter of a cylinder of height $L = 4$ and radius $r = 1$. The geometry is shown in Figure 19. We define the functions $g_{\phi,1}(\phi) = (1 - \cos\phi)(1 - \sin\phi)$, $g_{\phi,2} = (\cos\phi + \sin\phi - 4\sin\phi\cos\phi)$ and $g_z(z) = \sin(\frac{\alpha\pi z}{L})$ in cylindrical coordinates (r, ϕ, z). We choose the exact solution

$$u(\phi, z) = \beta g_{\phi,1}(\phi) g_z(z), \tag{5.3.1}$$

which yields the source function

$$f(\phi, z) = \beta g_z(z) \left(\frac{\alpha^2\pi^2 g_{\phi,1}(\phi)}{L^2} - g_{\phi,2}(\phi) \right), \tag{5.3.2}$$

with $\alpha = 3$ and $\beta = \frac{1}{(3/2 - \sqrt{2})}$. We use NURBS basis functions of order $p = 2$ in both parametric directions and we impose homogeneous Dirichlet boundary conditions. The exact and numerical solutions are shown in Figure 20. We also perform convergence analysis which is reported in Figure 21. The optimal rates of convergence are confirmed in this case too.

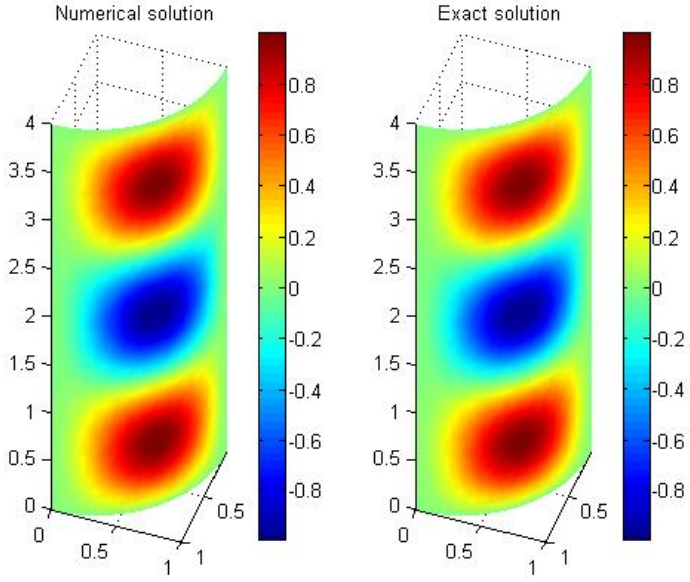

Figure 20: Exact and numerical solution of the Laplace-Beltrami problem for a quarter cylinder in 3D, using isogeometric analysis with NURBS basis functions of order $p = 2$. The number of elements in the refined mesh is $nel = 144$ with number of degrees of freedom $ndof = 625$.

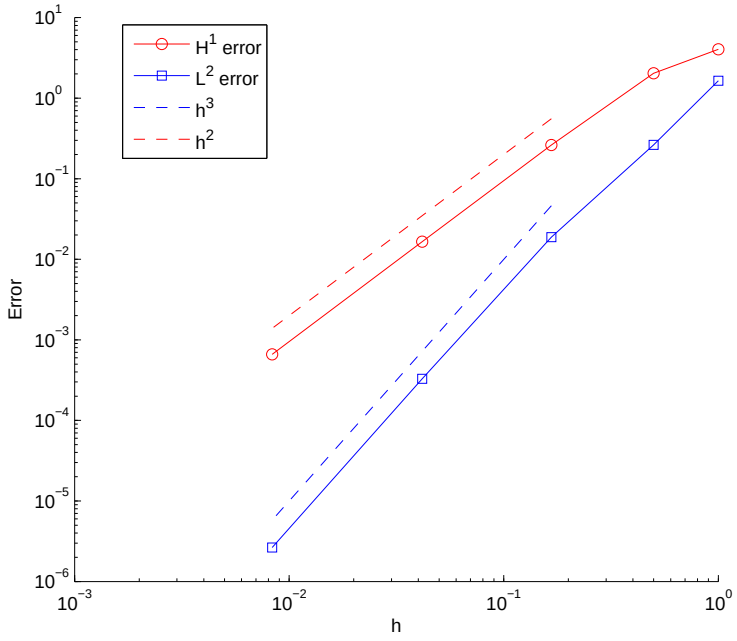

Figure 21: Plot of the L^2 and H^1 error norms for different mesh sizes h, for the Laplace-Beltrami problem in 3D using IGA. The optimal rates of convergence are confirmed.

5.4 BIHARMONIC PROBLEM FOR A SURFACE IN 3D

We solve the Dirichlet biharmonic problem 2.5.3 for the quarter cylinder with almost the same geometry (Figure 19) as in the Laplace-Beltrami problem. The only difference is that we choose the height of the cylinder $L = 8$. We use the same exact solution which yields the source function

$$f(\phi, z) = \frac{\beta g_z(z)\left(2\pi^4\alpha^4+\left(\pi^2\alpha^2+4L^2\right)^2\sin(2\phi)-2\left(\pi^2\alpha^2+L^2\right)^2(\sin(\phi)+\cos(\phi))\right)}{2L^4}.$$

We use NURBS basis functions of order $p = 3$ with C^1 inter-element continuity. The exact and numerical solutions are shown in Figure 22. We also check the convergence rates of the L^2,H^1 and H^2 error norms and we report the results in Figure 23.

Remark 5.4.1. *The biharmonic problem is a fourth order PDE and as such the approximate solution u_h must lie in the space $H^2(\Sigma)$. Therefore at least C^1 global continuity of basis functions is necessary. This requires special methods in FEM but is easily obtained in IGA.*

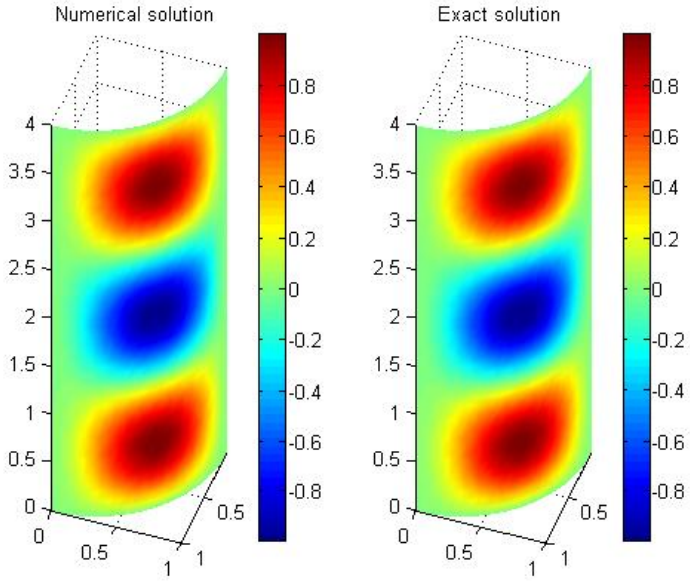

Figure 22: Exact and numerical solution of the Dirichlet biharmonic problem for a quarter cylinder in 3D, using isogeometric analysis with NURBS basis functions of order $p = 3$. The number of elements in the refined mesh is $nel = 144$ with number of degrees of freedom $ndof = 676$.

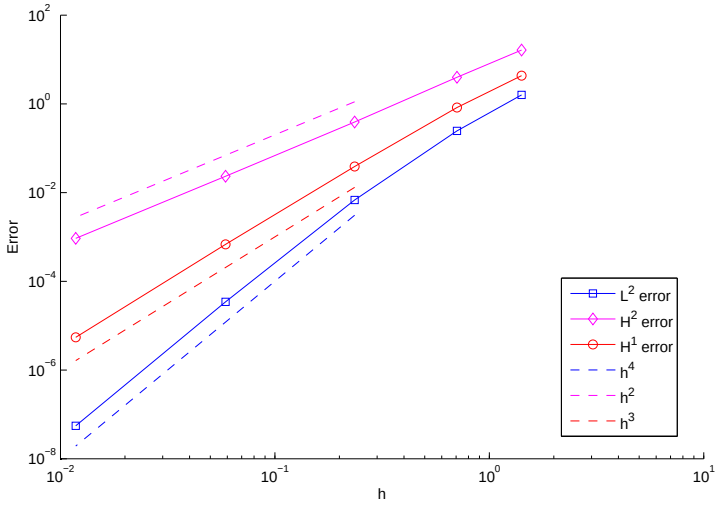

Figure 23: Plot of the L^2, H^1 and H^2 error norms for different mesh sizes h, for the Dirichlet biharmonic problem in 3D using IGA. The optimal rates of convergence are confirmed.

5.5 TIME-DEPENDENT ADVECTION-DIFFUSION PROBLEM

We solve the time-dependent advection-diffusion problem (2.5.12) for the cylinder with radius $r = 1$ and height $L = 6$. The geometry is depicted in Figure 24. For the advection field we choose

$$\mathbf{b}(\phi, z) = \frac{V_0}{\sqrt{1+\alpha^2}}(\widehat{\phi} + \alpha\widehat{z}), \tag{5.5.1}$$

with $V_0 = 1.5, \alpha = 0.4$. For the source function we choose

$$f(\phi, z) = e^{-b\left((\phi-\phi_0)^2+(z-z_0)^2\right)}, \tag{5.5.2}$$

with $b = 100$, $\phi_0 = \pi$, $z_0 = 1.5$. We set Dirichlet boundary conditions on Γ_D and we use NURBS basis functions of order $p = 3$ and C^0 global continuity. We first refine the mesh so that we have nel $= 3600$ and ndof $= 32,851$ and then we use the Crank-Nicholson method for the time stepping with $\Delta t = 0.01$.

We first solve the problem when $\mu = 1$, in which case diffusion is dominant. The solution at times $t = 0.2, t = 2, t = 4$ is shown in Figure 25. We see that the solution is smooth. We then choose $\mu = 10^{-5}$ so that advection dominates and the local Péclet numbers become of the order of $\mathcal{P}_{e_e} \sim 10^3$. The solution at times $t = 0.2, t = 2, t = 6$ is shown in Figure 26. One can see that instabilities have formed throughout the computational domain. Next, we use SUPG stabilization and we report the results in Figure 27. We see that SUPG has improved the stability of the solution throughout the domain.

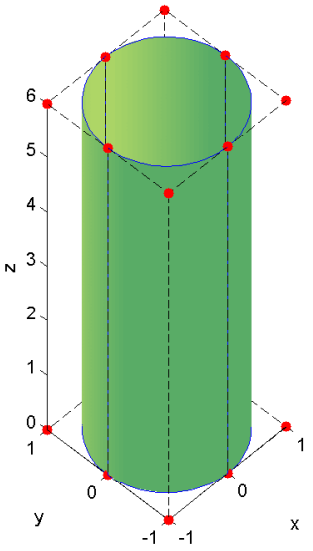

Figure 24: NURBS geometry of cylinder: exact representation, control points and control net in the coarsest level of discretization. There are four elements and eighteen control points. Note that there are two pairs of coinciding control points.

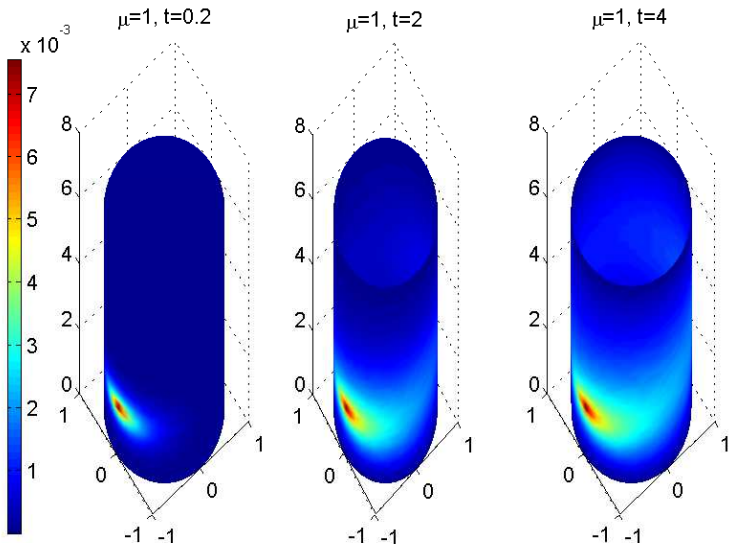

Figure 25: Solution of diffusion dominated time-dependent advection-diffusion problem at times $t = 0.2, t = 2, t = 4$. Diffusion is dominant with a slight transport by the advective field.

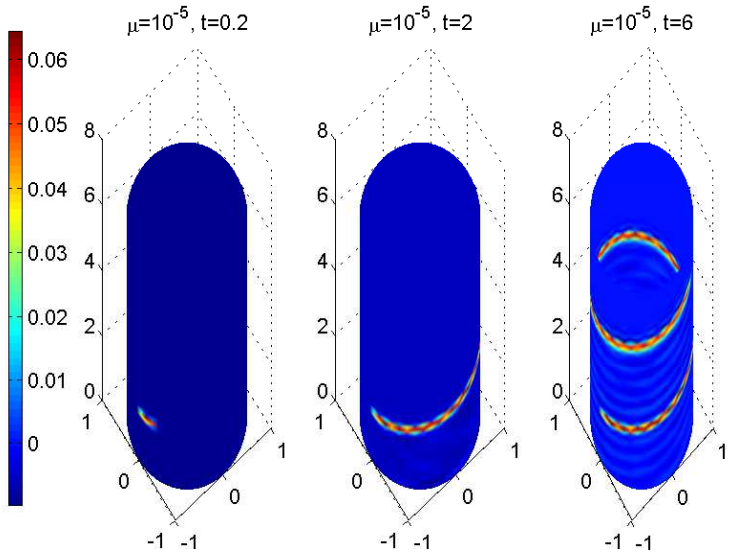

Figure 26: Solution of advection dominated time-dependent advection-diffusion problem at times $t = 0.2, t = 2, t = 6$. The solution is transported across the advective field. Instabilities form due to the high Péclet numbers.

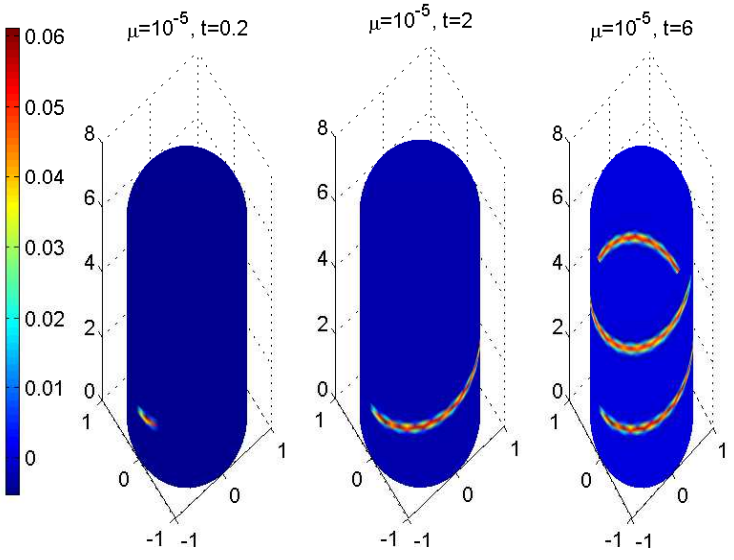

Figure 27: Solution of advection dominated time-dependent advection-diffusion problem with SUPG stabilization at times $t = 0.2, t = 2, t = 6$. The solution is transported across the advective field. Instabilities are contained with the use of SUPG stabilization.

6 CONCLUSIONS

6.1 CONCLUSIVE REMARKS

We have reviewed, implemented and analyzed the cut finite element method and isogeometric analysis for the solution of partial differential equations on manifolds. The numerical solutions have in all cases, except the unstabilized cut finite element method, agreed with the theoretical solutions. Furthermore, the convergence of the error norms has been in agreement with the theoretical a priori error estimates.

We have shown that the cut finite element method can be ill-conditioned in general but it becomes a reliable method when using stabilization. We can say that the cut FEM is well suited to the implicit representation method. However, when compared with IGA it falls behind in accuracy and efficiency for the example problem we tested. The cut FEM needs a higher number of degrees of freedom to produce the same order of error as IGA. We believe that in general, isogeometric analysis is inherently more suitable for the solution of PDEs on manifolds because it can represent many geometries exactly. While this is true in general, it is even more important for curved geometries because the error in the approximate representation can become large in these cases. Moreover, IGA is more suitable for the solution of higher order PDEs because increased continuity of the basis functions is readily available.

Nevertheless, there are issues with IGA too. As we mentioned, refinement in IGA is a global procedure that results in a lot of superfluous structure. Classic FEA has clearly the advantage here with proven refinement properties. Additionally, IGA requires that a parametrization of the physical object is available. This might be true if for example the object has been modeled beforehand using CAD, but it is not true in general. Also, the choice of parametric representation itself can influence the numerical analysis when there are artificial singularities or when the Jacobian of the parametrization is not sufficiently regular [SKBW10].

An area of application where the cut finite element method is more powerful than IGA is when one needs to model an evolving hypersurface. In these cases, where the shape and the topology might change in time, the implicit representation and the cut finite element method are more suitable choices.

In general, we can conclude that the two methods are better suited to different problems: the cut FEM is more suitable when

the parametrization is not available and for evolving interfaces, while IGA is more suitable when one has a CAD model readily available.

6.2 SUGGESTIONS AND FUTURE WORK

There are many areas of possible improvement. Starting from the cut finite element method, we can say that a more efficient implementation can be made using spatial data structures (quadtrees and octrees) and computational geometry algorithms. A more efficient code will allow the extension to the 3D case and the use of smaller mesh sizes. Additionally, we could apply the method to other problems and particularly to evolving hypersurfaces. Another interesting problem would be to use the cut FEM with NURBS basis functions which would result in some advantages such as increased continuity of the basis functions.

For isogeometric analysis, one could explore the use of T-splines and hierarchical refinement. Another topic that would be interesting is the use of periodic (or unclamped) knot vectors to be able to represent closed geometries more naturally. With regards to the implementation, potential improvements would be the implementation of optimal quadrature rules, the use of Bézier extraction and the use of other time stepping methods (e.g. generalized-α method). Finally, more complex problems could be examined that involve non-linear terms or more complicated geometries.

A

DEFINITIONS AND THEORETICAL DETAILS

A.1 DIFFERENTIAL GEOMETRY

Definition A.1.1. *A space is said to be an n-dimensional topological manifold if every point in it has a neighborhood homeomorphic to an open subset of* $\mathbb{R}^n$*. For every point* x *on the manifold there exists a map* $\alpha : U_\alpha \to \mathbb{R}^n$ *from an open neighborhood* U_α *of* x *such that* U_α *is homeomorphic to* $\alpha(U_\alpha)$*. The map* α *is called a chart and a collection of charts that covers the entire domain is called an atlas. If two charts* α *and* β *overlap in a region of the domain, then we can define the transition function* $\beta \circ \alpha^{-1} : \alpha(U_\alpha \cap U_\beta) \to \mathbb{R}^n$*. If all the transition functions are r-times continuously differentiable, the manifold is said to be a* C^r *differentiable manifold.*

Definition A.1.2. *A Riemannian manifold* M *is a manifold equipped with a metric* g*. The metric smoothly defines the scalar product of tangent vectors on the tangent space* $T_p M$ *for every point* $p \in M$*.*

Definition A.1.3. *An immersion* $f : M \to N$ *is a differentiable map between differentiable manifolds whose derivative is everywhere injective.*

Definition A.1.4. *If* M *and* N *are differentiable manifolds and* $\dim N - \dim M = 1$ *and if an immersion* $f : M \to N$ *has been defined, then* $f(M)$ *is a hypersurface in* N*.*

Definition A.1.5. *An embedding is defined to be an injective immersion which is homeomorphic onto its image.*

A.2 FUNCTIONAL ANALYSIS

Definition A.2.1. *A Hilbert space is a complete inner product space, that is an inner space equipped with an inner product and having the property that every Cauchy sequence is convergent.*

Definition A.2.2. *The* $L^2(\Sigma)$ *function space is a Hilbert space and is defined as*

$$L^2(\Sigma) = \{v : \Sigma \to \mathbb{R} \,:\, \|v\|_{L^2(\Sigma)} < \infty\}, \tag{A.2.1}$$

where

$$\|v\|_{L^2(\Sigma)} = \left(\int_\Sigma |v|^2 \, dx \right)^{1/2}. \tag{A.2.2}$$

We denote the L^2 *inner product as* $(u, v)_\Sigma = (u, v)_{L^2(\Sigma)} = \int_\Sigma uv \, dx$*.*

Definition A.2.3. *The* $H^1(\Sigma)$ *Hilbert space is defined as*

$$H^1(\Sigma) = \{v \in L^2(\Sigma) \,:\, \nabla v \in L^2(\Sigma)\}, \tag{A.2.3}$$

with norm

$$\|v\|^2_{H^1(\Sigma)} = \|v\|^2_{L^2(\Sigma)} + \|\nabla v\|^2_{L^2(\Sigma)}. \tag{A.2.4}$$

Definition A.2.4. *The condition number of a regular matrix* A *satisfies* $\kappa(A) \geqslant 1$ *and is defined as*

$$\kappa(A) = \|A\|\|A^{-1}\|, \tag{A.2.5}$$

where the operator norm is defined as

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}. \tag{A.2.6}$$

For a normal matrix the condition number is $\kappa(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$ *where* $\lambda(A)$ *is an eigenvalue of* A.

A.3 FINITE ELEMENT METHOD

Definition A.3.1. *Given a continuous function* f*, we define its continuous piecewise linear interpolant* $\pi f \in V_h$ *on a mesh* K *as*

$$\pi f = \sum_{i=1}^{n} f(N_i)\phi_i, \tag{A.3.1}$$

where N_i *are the nodes of the mesh and* ϕ_i *are the nodal basis functions.*

Definition A.3.2. *A finite element consists of a triplet:*

- *A polygon* $K \subset \mathbb{R}^n$.
- *A polynomial function space* P *on* K.
- *A set of* $n = \dim P$ *linear functionals* $L_i(\cdot)$, $i = 1, \ldots, n$ *defining the degrees of freedom.*

B

CODE LISTINGS

B.1 MATLAB CODE FOR THE CUT FINITE ELEMENT METHOD

B.1.1 *Main program*

```
%% Definitions
[p,t] = rectmesh(-2,-2,2,2,0.1); % Define regular rectangular
    mesh
xc=0;
yc=0;
radius=1;
s = @(x) sqrt((x(1,:)-xc).^2+(x(2,:)-yc).^2)-radius; % Define
    signed distance function
grads = @(x) [x(1)-xc;x(2)-yc]./(sqrt((x(1)-xc).^2+(x(2)-yc)
    .^2));% Gradient of signed distance function
fp = @(r,phi) 108*sin(3*phi); % Source function in polar
    coordinates
f = @(x) fp(radius,atan2(x(2),x(1))); % Source function in
    cartesian coordinates
u = @(x) 12*(3*x(1,:).^2.*x(2,:)-x(2,:).^3); % Exact solution
gradu = @(x) [72*x(1,:).*x(2,:),12*(3*x(1,:).^2-3*x(2,:).^2)
    ]; % Gradient of exact solution

%% Preliminary calculations
[pin, ti, ip] = findIntersect(p, t, s, true); % Get sub-mesh
    and interection points
[r, rn, rl] = findVectors(ip); % Find tangent and normal
    vectors

%% Finite element assembly
A = AssembleStiff(pin, ti, r, rn, rl, ip, false); % Assemble
    stiffness matrix
lv = LoadVec(pin, ti, r, rl, ip, f, s, grads, 5); % Assemble
    load vector
C = AssembleConstraint(pin, ti, r, rl, ip); % Assemble
    constraint matrix
J = Stabilization(pin, ti); % Assemble stabilization term

%% Augmented system (Lagrange multiplier method)
AA = [A C';C 0];
AAS = [A+J C';C 0]; % Stabilized
bb = [lv;0];

%% Solution and analysis
eigv = eig(AA); % Eigenvalues of stiffness matrix
eigvs = eig(AAS); % Eigenvalues of stabilized stiffness
    matrix
```

```
condn = eigv(end)/eigv(3); % Effective condition number
condns = eigvs(end)/eigvs(2); % Stabilized eff. cond. number
uh = AAS\bb;
uh = uh(1:end-1);
l2e = L2Error(pin, ti, r, rl, ip, uh, u, s, grads); %
    Calculate error in L^2 norm
h1e = H1Error(pi, ti, r, rl, ip, uh, u, gradu, s, grads, l2e)
    ;  % Calculate error in H^1 norm

%% Visualization
% Get solutions at intersection points
[femsol, exsol] = EvalSol(pin, ti, ip(:,1:end-1), uh, u);
femsol = [femsol;femsol(1)];
figure(2)
subplot(1,2,1)
plot3(ip(1,:),ip(2,:),femsol,'b')
grid on
subplot(1,2,2)
tt = linspace(0,2*pi,100);
plot3(xc+radius*cos(tt),yc+radius*sin(tt),12*(3*(cos(tt))
    .^2.*(sin(tt))-(sin(tt).^3)),'r');
fprintf('L2 error: %f\n',l2e);
fprintf('L2 error: %f\n',h1e);
fprintf('Nel: %f\n',size(ti,2));
fprintf('Ndof: %f\n',length(uh));
```

B.1.2 *Intersection routine*

```
function [pi, ti, ip] = findIntersect(p, t, foo, plb)
nel = size(t,2); % Number of elements
nnodes = size(p,2); % Number of nodes
l = sqrt(nnodes);
x = p(1,:); % Node coordinates
y = p(2,:);
pif = interpol(foo, p); % Interpolate level set function
XX = reshape(x,l,l);
YY = reshape(y,l,l);
ZZ = reshape(pif,l,l);
if plb
    figure(1)
    hold on
    mesh(XX,YY,zeros(l,l));
    contour(XX,YY,ZZ,[0 0],'r');
    hold off
end
%% Find intersection points and which elements are
    intersected
ip = contourc(XX(1,:),YY(:,1),ZZ,[0 0]); % Zero level
ip = ip(:,2:end);
ipm = (ip(:,1:end-1) + ip(:,2:end))/2; % Midpoints
xe = p(1,t); % Element node coordinates
ye = p(2,t);
```

```
xe = reshape(xe,4,nel);
ye = reshape(ye,4,nel);
xe = xe(1:2,:);
ye = ye(2:3,:);
nti = length(ipm); % Number of intersected elements
intElem = zeros(nti,1);
idx=1:nel;
for i=1:nti
    idxx = idx(sum(xe < ipm(1,i),1)==1);
    idxy = idx(sum(ye < ipm(2,i),1)==1);
    intElem(i) = idxx(ismember(idxx,idxy));
end
%% New mesh matrices
[nodesi,~,ti] = unique(t(:,intElem));
ti = reshape(ti,4,[]);
pin = p(:,nodesi);
```

B.1.3 *Stiffness matrix assembly routine*

```
function A = AssembleStiff(pin, ti, r, rn, rl, ip, symb)
%% Assemble stiffness matrix
nti = size(ti,2);
npi = size(pin,2);
A = sparse(npi,npi);
inner = @(x) x'*x;
if symb
    syms xsym ysym tsym real
end
for K=1:nti
    P = eye(2) - rn(:,K)*rn(:,K)'; % Projection operator
    loc2glb = ti(:,K);
    xe = pin(1,loc2glb);
    ye = pin(2,loc2glb);
    % Get basis functions
    [~, ~, b, c, d] = Bilinear(xe,ye);
    if symb
        % Define symbolic variables
        rts = ip(:,K) + r(:,K) * tsym;
        basisGradS = [b+d*ysym,c+d*xsym]';
        integrandS = (P*basisGradS)'*(P*basisGradS);
        integrandS = subs(integrandS,[xsym;ysym],rts)*rl(K);
        AK = double(int(integrandS,0,1));
    else
        rt = @(tq) ip(:,K) + r(:,K) * tq; % Parametrization
        term = @(tq) P*BilinearGrad(rt(tq),b,c,d);
        integrand = @(tq) inner(term(tq));
        AK = rl(K)*GaussLegendre(integrand,0,1,2);
    end
    A(loc2glb,loc2glb) = A(loc2glb,loc2glb) + AK;
end
```

B.2 MATLAB CODE FOR ISOGEOMETRIC ANALYSIS

B.2.1 *Main program for the Laplace-Beltrami problem in 2D*

```
%%Definitions
xc=0;
yc=0;
ra=1;

fp = @(r,phi) 108*sin(3*phi); % Source function in polar
    coordinates
f = @(x,y,z) fp(1,atan2(y,x)); % Source function in cartesian
     coordinates

u = @(x,y,z) 12*(3*x.^2.*y-y.^3); % Exact solution
%Exact tangential gradient
gradu = @(x,y,z) cat(1,reshape(36*x.*y.*(4*y.^2-1),[1 size(x)
    ]),reshape(36*(1-5*y.^2+4*y.^4),[1 size(x)]),reshape(
    zeros(size(x)),[1 size(x)]));

c_diff = @(x,y,z) ones(size(x)); % Coefficient

%% Geometry and problem set-up
geo = nrbcirc(ra,[xc,yc]);
geometry = geo_load(geo);
regularity =0;
nsub=3;
[rknots, zeta, nknots] = kntrefine (geometry.nurbs.knots,
    nsub-1, geometry.nurbs.order-1, regularity);
h = max(diff(zeta));
nel = numel(zeta)-1;
nurbs = nrbkntins (geometry.nurbs, nknots);
geometry = geo_load (nurbs);
nquad = geometry.nurbs.order;

%% Plot geometry and control points
figure(1)
nrbplot(geometry.nurbs,20);
nrbctrlplot(geometry.nurbs);

%% Construct msh structure
rule     = msh_gauss_nodes (nquad);
[qn, qw] = msh_set_quad_nodes ({zeta}, rule);
msh      = msh_1d ({zeta}, qn, qw, geometry, 'boundary',
    false);

%% Construct space structure
space  = sp_nurbs_1d (geometry.nurbs, msh, 'ccp', true);

%% Assemble the matrices
stiffmat = op_gradu_gradv_tp (space, space, msh, c_diff);
stiffmat(end:end)=stiffmat(1,1);
rhs       = op_f_v_tp (space, msh, f);
```

```
rhs(end) = rhs(1);
condmat = op_f_v_tp(space, msh, c_diff);
condmat(end)=condmat(1);

%% Set up augmented system (lagrange multiplier method)
stiffmat = [stiffmat condmat;condmat' 0];
rhs = [rhs;0];

%% Solution
sol = stiffmat\rhs;
assert(sol(end)<1e-9)
sol = sol(1:end-1);

pts = linspace(0,1,100);
vtk_pts = {pts};
[eu,F] = sp_eval(sol,space,geometry,vtk_pts);
[X, Y]  = deal (squeeze(F(1,:,:)), squeeze(F(2,:,:)));
figure(2)
subplot (2,1,2)
plot3 (X, Y, eu,'r')
title ('Numerical solution'), axis tight
subplot (2,1,1)
plot3 (X, Y, u (X,Y))
title ('Exact solution'), axis tight
[h1e,l2e] = sp_h1_error(space, msh, sol, u, gradu);
```

B.2.2 *Main program for the biharmonic problem in 3D*

```
%% Parameters
xa=0;
ya=0;
za=0;
L=8;
radius=1;
angle=pi/2;
alpha = 3;
beta = 1/(1.5-sqrt(2));

%% Source function
gphi1 = @(phi) (1-cos(phi)).*(1-sin(phi));
gphi2 = @(phi) (cos(phi)+sin(phi)-4*sin(phi).*cos(phi));
gz = @(z) sin(alpha*pi*z/L);
fcyl = @(r,phi,z) (1/(2*L^4)).*beta.*gz(z).*(2*alpha^4*pi
    ^4-2*(L^2+alpha^2*pi^2)^2*(cos(phi)+sin(phi)) +...
    (4*L^2+alpha^2*pi^2).^2*sin(2*phi));
f = @(x,y,z) fcyl(radius, atan2(y,x), z );


%% Exact solution
ucyl = @(r,phi,z) beta*gphi1(phi).*gz(z);
uex = @(x,y,z) ucyl( radius, atan2(y,x) ,z);
```

```
%% Exact tangential gradient

gradx = @(x,y,z) beta.*y.*(x+(y-1).*(1+2*y)).*sin(alpha*pi*z/
    L);
grady = @(x,y,z) beta.*(-1+y.^2+x.*(1+y-2*y.^2)).*sin(alpha*
    pi*z/L);
gradz = @(x,y,z) alpha*beta*pi*(x-1).*(y-1).*cos(alpha*pi*z/L
    )/L;
graduex = @(x,y,z) cat(1,reshape(gradx(x,y,z),[1 size(x)]),
    reshape(grady(x,y,z),[1 size(x)]),reshape(gradz(x,y,z),[1
     size(x)]));

%% Exact tangential Laplacian
laplcyl = @(r,phi,z) -gz(z).*((((alpha^2)*pi^2)/L^2).* gphi1(
    phi) - gphi2(phi)).*beta;
lapluex = @(x,y,z) laplcyl(radius, atan2(y,x), z );

c_diff = @(x,y,z) ones(size(x));

%% Geometry and problem set-up
geo = nrbcylind(L,radius,[xa,ya,za],0,pi/2);
geometry = geo_load(geo);
degree = [3 3];
regularity = [1 1];
nsub=7;
degelev  = max (degree - (geometry.nurbs.order-1), 0);
nurbs    = nrbdegelev (geometry.nurbs, degelev);
[rknots, zeta, nknots] = kntrefine (nurbs.knots, [nsub-1 nsub
    -1], nurbs.order-1, regularity);
part1 = diff(zeta{1}).^2;
part2 = diff(zeta{2}).^2;
h = sqrt(max(part1)+max(part2));
nurbs = nrbkntins (nurbs, nknots);
geometry = geo_load (nurbs);
nquad = geometry.nurbs.order+3;
figure(1)
nrbplot(geometry.nurbs,[20 20]);
nrbctrlplot(geometry.nurbs);

%% Construct msh structure
rule     = msh_gauss_nodes (nquad);
[qn, qw] = msh_set_quad_nodes (zeta, rule);
msh      = msh_2d (zeta, qn, qw, geometry, 'boundary', true,
    'der2', true);
nel = msh.nel

%% Construct space structure
space  = sp_nurbs_2d (geometry.nurbs, msh, 'ccp', false);

% Assemble the matrices
stiffmat = op_laplaceu_laplacev_tp (space, space, msh, c_diff
    );
rhs       = op_f_v_tp (space, msh, f);
```

```
u = zeros (space.ndof, 1);

%% Enforce boundary conditions
drchlt_dofs_u = []; drchlt_dofs_r = [];
for iside = [1 2]
  drchlt_dofs_u = union (drchlt_dofs_u, space.boundary(iside)
      .dofs);
  drchlt_dofs_r = union (drchlt_dofs_r, space.boundary(iside)
      .adjacent_dofs);
end
for iside = [3 4]
  drchlt_dofs_u = union (drchlt_dofs_u, space.boundary(iside)
      .dofs);
end
drchlt_dofs = union (drchlt_dofs_u, drchlt_dofs_r);

int_dofs = setdiff (1:space.ndof, drchlt_dofs);
rhs(int_dofs) = rhs(int_dofs) - stiffmat(int_dofs,
    drchlt_dofs)*u(drchlt_dofs);

%% Solve the linear system
u(int_dofs) = stiffmat(int_dofs, int_dofs) \ rhs(int_dofs);

pts = linspace(0,1,50);
vtk_pts = {pts,pts};
[eu,F] = sp_eval(u,space,geometry,vtk_pts);
[X, Y, Z]  = deal (squeeze(F(1,:,:)), squeeze(F(2,:,:)),
    squeeze(F(3,:,:)));
figure(2)
subplot (1,2,1)
surf (X, Y, Z, eu)
title ('Numerical solution'), axis tight
subplot (1,2,2)
surf (X, Y, Z, uex (X,Y,Z))
title ('Exact solution'), axis tight
[h2en(nn),h1en(nn),l2en(nn)] = sp_h2_error(space, msh, u, uex
    , graduex, lapluex);
```

BIBLIOGRAPHY

[BBdVC+06] Y Bazilevs, L Beirao da Veiga, JA Cottrell, TJR Hughes, and G Sangalli, *Isogeometric analysis: approximation, stability and error estimates for h-refined meshes*, Mathematical Models and Methods in Applied Sciences **16** (2006), no. 07, 1031–1090. (Cited on page 30.)

[BCC+10] Y Bazilevs, VM Calo, JA Cottrell, JA Evans, TJR Hughes, S Lipton, MA Scott, and TW Sederberg, *Isogeometric analysis using t-splines*, Computer Methods in Applied Mechanics and Engineering **199** (2010), no. 5, 229–263. (Cited on page 29.)

[Ber03] Marcel Berger, *A panoramic view of riemannian geometry*, Springer, 2003. (Cited on pages 1 and 10.)

[Bro08] Alexander M. Bronstein, *Numerical geometry of non-rigid shapes*, Monographs in computer science, Springer, New York, NY, 2008. (Cited on page 1.)

[BS08] Susanne Brenner and Larkin Ridgway Scott, *The mathematical theory of finite element methods*, Springer, New York [u.a.], 2008 (English). (Cited on pages 1 and 15.)

[DDEH09] K. Deckelnick, G. Dziuk, C. M. Elliott, and C. J. Heine, *An h-narrow band finite-element method for elliptic equations on implicit surfaces*, IMA Journal of Numerical Analysis **30** (2009), 351–376. (Cited on page 2.)

[DE13] Gerhard Dziuk and Charles M. Elliott, *Finite element methods for surface PDEs*, Acta Numerica **22** (2013), 289–396. (Cited on page 2.)

[DFRV11] C. De Falco, A. Reali, and R. Vázquez, *GeoPDEs: a research tool for isogeometric analysis of PDEs*, Advances in Engineering Software **42** (2011), no. 12, 1020–1034. (Cited on page 3.)

[DQ13] L. Dedé and A. Quarteroni, *Isogeometric analysis for second order partial differential equations on surfaces*, Tech. report, MOX Report No. 06/2013, 2013. (Cited on pages 2 and 31.)

[dVBRS11] L Beirao da Veiga, A Buffa, J Rivas, and G Sangalli, *Some estimates for h–p–k-refinement in isogeometric analysis*, Numerische Mathematik **118** (2011), no. 2, 271–305. (Cited on page 30.)

[Dzi88] Gerhard Dziuk, *Finite elements for the beltrami operator on arbitrary surfaces*, Lecture Notes in Mathematics **1357** (1988), 142–155. (Cited on page 2.)

[ES10] Charles M. Elliott and Björn Stinner, *Modeling and computation of two phase geometric biomembranes using surface finite elements*, Journal of Computational Physics **229** (2010), no. 18, 6585–6612. (Cited on page 1.)

[GT12] KPS Gahalaut and SK Tomar, *Condition number estimates for matrices arising in the isogeometric discretizations*, RICAM report **23** (2012), no. 2012, 1–38. (Cited on page 31.)

[HCB05] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs, *Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement*, Computer Methods in Applied Mechanics and Engineering **194** (2005), 4135–4195. (Cited on page 2.)

[HCB09] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs, *Isogeometric analysis: Toward integration of CAD and FEA*, John Wiley & Sons, 2009. (Cited on pages 2 and 26.)

[HLZ13] Peter Hansbo, Mats G. Larson, and Sara Zahedi, *Characteristic cut finite element methods for convection-diffusion problems on time dependent surfaces*, Tech. Report 2013-004, Uppsala University, Division of Scientific Computing, 2013. (Cited on page 17.)

[HRS10] T.J.R. Hughes, A. Reali, and G. Sangalli, *Efficient quadrature for NURBS-based isogeometric analysis*, Computer Methods in Applied Mechanics and Engineering **199** (2010), no. 5-8, 301–313. (Cited on page 32.)

[JL04] Ashley J James and John Lowengrub, *A surfactant-conserving volume-of-fluid method for interfacial flows with insoluble surfactant*, Journal of computational physics **201** (2004), no. 2, 685–722. (Cited on page 1.)

[LB12] Mats G. Larson and Fredrik Bengzon, *The finite element method: theory, implementation, and practice*,

Springer, New York, 2012. (Cited on pages 1 and 15.)

[LL13] K. Larsson and M. G. Larson, *A continuous/discontinuous Galerkin method and a priori error estimates for the biharmonic problem on surfaces*, arXiv:1305.2740, May 2013. (Cited on page 17.)

[MAT13] MATLAB, *version 8.1 (r2013a)*, The MathWorks Inc., Natick, Massachusetts, 2013. (Cited on page 3.)

[MLL12] A. Massing, M. G. Larson, and A. Logg, *Efficient implementation of finite element methods on non-matching and overlapping meshes in 3D*, arXiv:1210.7076, October 2012. (Cited on page 24.)

[NSBR12] Vinh Phu Nguyen, Robert N. Simpson, Stéphane Bordas, and Timon Rabczuk, *An introduction to isogeometric analysis with matlab implementation: FEM and XFEM formulations*, arXiv:1205.2129, 2012. (Cited on page 26.)

[OF03] Stanley Osher and Ronald Fedkiw, *Level set methods and dynamic implicit surfaces*, vol. 153, Springer, 2003. (Cited on page 1.)

[OR10] Maxim A. Olshanskii and Arnold Reusken, *A finite element method for surface pdes: matrix properties*, Numerische Mathematik **114** (2010), no. 3, 491–520 (English). (Cited on pages 17 and 21.)

[ORG09] Maxim A. Olshanskii, Arnold Reusken, and Jörg Grande, *A finite element method for elliptic equations on surfaces*, SIAM Journal on Numerical Analysis **47** (2009), no. 5, 3339–3358. (Cited on pages 2, 17, and 21.)

[ORX12] Maxim A. Olshanskii, Arnold Reusken, and Xianmin Xu, *A volume mesh finite element method for PDEs on surfaces*, (J. Eberhardsteiner et.al., ed.), European Congress on Computational Methods in Applied Sciences and Engineering, 2012. (Cited on page 17.)

[ORX13a] M. A. Olshanskii, A. Reusken, and X. Xu, *A stabilized finite element method for advection-diffusion equations on surfaces*, arXiv:1301.3741, January 2013. (Cited on page 17.)

[ORX13b] M. A. Olshanskii, A. Reusken, and X. Xu, *An Eulerian space-time finite element method for diffusion*

problems on evolving surfaces, arXiv:1304.6155, April 2013. (Cited on page 17.)

[ORX13c] M. A. Olshanskii, A. Reusken, and X. Xu, *On surface meshes induced by level set functions*, arXiv:1301.3745, January 2013. (Cited on page 17.)

[Pre10] Andrew N. Pressley, *Elementary differential geometry*, Springer, 2010. (Cited on page 1.)

[PT97] Les Piegl and Wayne Tiller, *The nurbs book*, Springer-Verlag, 1997. (Cited on pages 2 and 26.)

[Rog01] David F Rogers, *An introduction to NURBS with historical perspective*, Morgan Kaufmann Publishers, San Francisco, 2001 (English). (Cited on pages 2 and 26.)

[RWP06] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke, *Laplace-beltrami spectra as 'shape-dna' of surfaces and solids*, Comput. Aided Des. **38** (2006), no. 4, 342–366. (Cited on pages 1 and 12.)

[RWSN09] Martin Reuter, Franz-Erich Wolter, Martha Shenton, and Marc Niethammer, *Laplace-beltrami eigenvalues and topological features of eigenfunctions for statistical shape analysis*, Computer-Aided Design **41** (2009), no. 10, 739 – 755. (Cited on pages 1 and 12.)

[SBV+11] Michael A. Scott, Michael J. Borden, Clemens V. Verhoosel, Thomas W. Sederberg, and Thomas J. R. Hughes, *Isogeometric finite element data structures based on bézier extraction of t-splines*, International Journal for Numerical Methods in Engineering **88** (2011), no. 2, 126–156. (Cited on page 31.)

[Shu01] Mikhail A. Shubin, *Pseudodifferential operators and spectral theory*, Springer, 2001. (Cited on page 43.)

[SKBW10] R. Schmidt, J. Kiendl, K.-U. Bletzinger, and R. Wüchner, *Realization of an integrated structural design process: analysis-suitable geometric modelling and isogeometric analysis*, Computing and Visualization in Science **13** (2010), no. 7, 315–330 (English). (Cited on page 53.)

[TDQ13] A Tagliabue, L Dedè, and A Quarteroni, *Isogeometric analysis and error estimates for high order partial differential equations in fluid dynamics*, Tech. report, MOX Report, 2013. (Cited on page 31.)

[TLNC10] A. Tveito, H.P. Langtangen, B.F. Nielsen, and X. Cai, *Elements of scientific computing*, Texts in Computational Science and Engineering, Springer, 2010. (Cited on page 24.)

[VGJS11] A-V Vuong, Carlotta Giannelli, Bert Jüttler, and Bernd Simeon, *A hierarchical approach to adaptive local refinement in isogeometric analysis*, Computer Methods in Applied Mechanics and Engineering **200** (2011), no. 49, 3554–3567. (Cited on page 29.)

[VHS10] A.-V. Vuong, Ch. Heinrich, and B. Simeon, *ISOGAT: a 2D tutorial MATLAB code for isogeometric analysis*, Computer Aided Geometric Design **27** (2010), no. 8, 644–655. (Cited on page 26.)

[WRP07] F-E Wolter, M Reuter, and N Peinecke, *Geometric modeling for engineering applications*, Encyclopedia of Computational Mechanics (2007). (Cited on page 1.)