PDE-Based Parameterization Techniques for Isogeometric Analysis Applications

Hinz, Jochen

DOI
10.4233/uuid:6ae552a0-7425-4d7c-afd5-5b7a42a10f27

Publication date
2020

Document Version
Final published version

Citation (APA)
Hinz, J. (2020). *PDE-Based Parameterization Techniques for Isogeometric Analysis Applications*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:6ae552a0-7425-4d7c-afd5-5b7a42a10f27

PDE-Based Parameterization Techniques for Isogeometric Analysis Applications

PDE-Based Parameterization Techniques for Isogeometric Analysis Applications

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Friday 17 July 2020 at 15:00 o'clock

by

Jochen Peter HINZ

Master of Science in Applied Physics and Applied Mathematics,
Delft University of Technology, the Netherlands.
Born in Kempen, Germany.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairman
Prof. dr. ir. C. Vuik,	Delft University of Technology, promotor
Dr. ir. M. Möller,	Delft University of Technology, copromotor
Independent members:	
Univ.-Prof. dr. B. Jüttler	Johannes Kepler Universität Linz, Austria
Prof. dr. T. Verstraete	Von Karman Institute for Fluid Dynamics, Belgium
Prof. dr. ir. E.H. van Brummelen	Eindhoven University of Technology
Univ.-Prof. S. Elgeti	Technische Universität Wien, Austria
Prof. dr. ir. A. W. Heemink	Delft University of Technology
Prof. dr. ir. K.I. Aardal	Delft University of Technology, reserve member

Keywords: Isogeometric Analysis, parameterization techniques, elliptic grid generation, matrix-free Newton-Krylov, mixed-FEM, THB-splines, twin-screw machine geometries

Printed by: ProefschriftMaken || proefschriftmaken.nl

Front & Back: Vera van Beek.

The author gratefully acknowledges the research funding which was partly provided by the *MOTOR* project that has received funding from the European Unions Horizon 2020 research and innovation program under grant agreement No 678727.

ISBN 978-94-6384-146-7

An electronic version of this dissertation is available at
`http://repository.tudelft.nl/`.

"All dissertation epigraphs can be trusted."

– Christopher Columbus

CONTENTS

SUMMARY

The aim of this dissertation is to introduce the concept of PDE-based parameterization using *Isogeometric Analysis* (IGA) techniques and to present first results. IGA is a numerical technique that employs the spline- and NURBS-based geometric modeling tools of *Computer Aided Geometric Design* (CAGD) as a basis for numerical simulation using the principles of *Finite Element Analysis* (FEA). This work proposes techniques that employ a parametric domain comprised of one (singlepatch) or several (multipatch) unit quadrilaterals and find an analysis-suitable mapping operator that parameterizes the physical domain. Here, the only input is a NURBS-based correspondence between the boundaries of the parametric and physical domain. To achieve this, this work adopts the principles of *Elliptic Grid Generation* (EGG), a PDE-based technique from classical meshing, and proposes discretizations that are suitable for an IGA-based computational workflow. The PDE-based problem formulation is motivated by the prospect of employing the same numerical technique for the geometrical as well as the computational aspects of the numerical simulation pipeline.

We provide means to generate robust initial guesses for the (Newton-based) iterative approach that lies at the heart of the computational algorithm. To enable its application in large-scale numerical simulations, we provide convenient memory-saving strategies such as matrix-free Newton-Krylov (singlepatch) and Schur complement Newton-Krylov (multipatch).

We further augment the methodology with the concept of *domain optimization*, which allows for precisely controlling the parametric properties of the PDE solution. This is accomplished through a coordinate transformation in the parametric domain. We combine the PDE-based approach with the principle of parameterization quality cost function optimization to choose an appropriate transformation. Hereby, operating in the parametric domain rather than directly on the geometry potentially improves robustness and efficiency.

For removing mapping degeneracies resulting from the truncation error, this work proposes a posteriori refinement strategies based on duality considerations. They are combined with an unstructured spline technology to enable computationally efficient local refinement. The a posteriori strategies substantially improve the robustness of an IGA-based numerical simulation framework. However, since defect correction is based on (local) refinement, improved robustness may come at the expense of higher computational costs.

The monolithic treatment of geometry and simulation is taken advantage of by merging the residuals corresponding to the state variable (e.g. temperature, pressure) and the mapping into a joint IGA-residual. This allows for deriving a symbolic expression for the gradient of a shape optimization cost function with a convoluted dependency on the geometry. The expression is then combined with a gradient-based algorithm that optimizes the shape of a domain.

To further validate the proposed techniques, we employ them for the fully automated parameterization of co- and counter-rotating twin-screw machine geometries in two and three spatial dimensions. Hereby, the methodology takes advantage of the differentiability of the PDE-problem with respect to the boundary correspondence. By regarding the mapping as a smooth function of the rotational angle, a database is hierarchically filled with planar parameterizations for a large number of discrete angles. Interpolation within the partially-filled database speeds up the computation of the remaining parameterizations. The spline-based mappings are used for the numerical simulation of molten polymers within twin-screw machine extruders. In this application, they are orthogonalized using a control mapping. The smooth dependency of the database on the rotational angle improves the numerical quality of volumetric twin-screw machine parameterizations, which are generated by selecting a large number of planar parameterizations and stacking them in the z-direction.
The techniques presented in this work demonstrate the potential advantages of a computational workflow that consolidates the principles of CAGD and FEA. By not only integrating FEA into CAGD-based design tools but also employing FEA techniques to solve a CAGD problem, this work pays particular attention to a bidirectional unification of both fields.

1

INTRODUCTION

Finite element analysis (FEA) has become an indispensable numerical technique for approximating solutions of boundary value problems (BVP) as they arise in most engineering applications. The technique is well-grounded in functional analysis [Red13]. It has been successfully applied to problems in, among others, structural analysis [Yan86, ZTNZ77], computational electromagnetism [BRI05], computational fluid dynamics [GR12] and heat transfer [LMTS96].
In this section, we discuss the geometrical aspects of FEA and the challenges they pose to automating engineering design and simulation workflows. Furthermore, we give a motivation and state the key objectives of this work.

1.1. MESH GENERATION

Assuming that the BVP is posed over a simply connected open domain $\Omega \subset \mathbb{R}^n$, $n \in \{2,3\}$, a preceding step to performing FEA is to subdivide Ω into a set $\mathcal{T}$ of mutually disjoint cells (e.g. triangles, quadrilaterals, hexahedrons), given no more than a parametric description of the boundary $\partial\Omega$. The $K_i \in \mathcal{T}$ are chosen such that

$$\mathrm{Int}\left(\bigcup_{K_i \in \mathcal{T}} \overline{K}_i\right) = \Omega_h$$

and $\partial\Omega_h$ is a collocation of $\partial\Omega$ (see Figure 1.1). Here, $\mathrm{Int}(\cdot)$ denotes the interior of a closed domain and $\overline{K}_i$ the closure of an open domain K_i, respectively. The process of selecting a suitable subdivision $\mathcal{T}$ is referred to as *meshing*, a subdiscipline of *Computer Aided Geometric Design* (CAGD). Typically, meshing strategies operate in the order $\partial\Omega \to \partial\Omega_h \to \mathcal{T}$.
The $K_i \in \mathcal{T}$ are referred to as *elements* and are characterized by their vertices $\mathbf{p}_j \in \mathbb{R}^n$. Since the $\mathbf{p}_j$ determine the properties of the (typically piecewise polynomial) bases that are compatible with $\mathcal{T}$, their choice has a profound impact on the properties of the subsequent FEA. Let each $K_i \in \mathcal{T}$ be the image of some reference element $\hat{K} \subset \mathbb{R}^m$ with $m \leq n$, under a mapping $\mathbf{F}_i : \hat{K} \to K_i$ taken from the linear span of a (typically Lagrangian

1

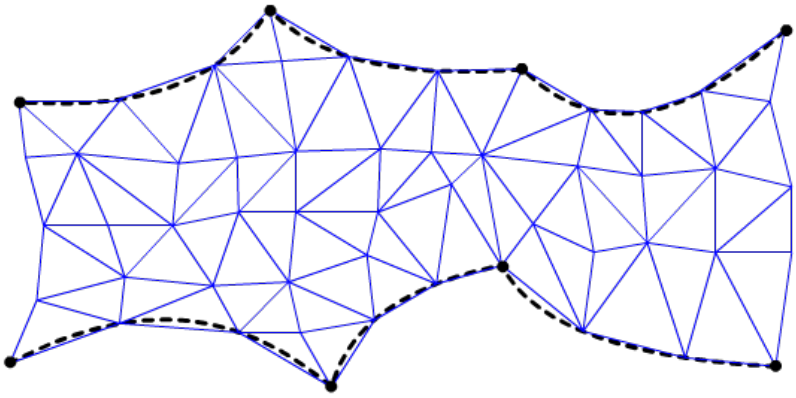


Figure 1.1: An unstructured triangulation of the interior of the physical domain Ω fenced off by (piecewise) NURBS-based parametric curves.

[War79]) polynomial basis. A *valid* mesh is one in which each $\mathbf{F}_i$ is a homeomorphism. A $K_i \in \mathscr{T}$ that fails to be homeomorphic to $\hat{K}$ is referred to as a *degenerate element* (see Figure 1.2). In practice, meshes can fail to be suitable for FEA (analysis-suitable) despite be-

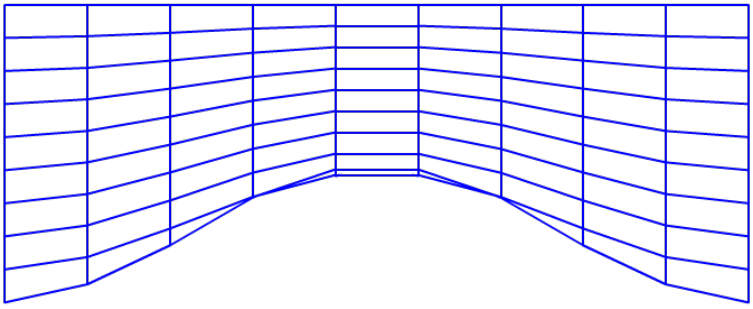


Figure 1.2: A degenerate structured quadrangulation. Here, elements near the southern boundary degenerate due to extreme skewness. A total of three elements fail to be homeomorphic to the unit rectangle. The mesh is not analysis-suitable.

ing nondegenerate. Hence, commonly-applied algorithms furthermore attempt to select $\mathscr{T}$ such that the $K_i \in \mathscr{T}$ have favorable numerical properties. Mesh quality metrics are largely based on heuristics but typically avoid near-degeneracies, reduce the *skewness* introduced by the deformations $\mathbf{F}_i$ and ensure that neighbouring cells vary smoothly in size [Knu01]. While some applications perform FEA with Lagrangian elements of polynomial order $p \geq 2$, in the vast majority of cases, the $K_i \in \mathscr{T}$ are piecewise linear transformations of $\hat{K}$, while the $K_i \in \mathscr{T}$ form a simplical complex in $\mathbb{R}^n$.
Meshes can be divided into two types: structured and unstructured. In the former case, the $K_i \in \mathscr{T}$ form a regular lattice while in the latter, the cells can have irregular connectivities. Clearly, the flexibility of unstructured meshes facilitates capturing complicated domains, while structured meshes have computational advantages, such as leading to

banded FEA matrices that allow for a more hardware-friendly implementation and parallelization of assembly and inversion [Dic79].

1.2. THE ROLE OF NUMERICAL TECHNIQUES IN COMPUTATIONAL ENGINEERING WORKFLOWS

Computational science and engineering (CSE) is a field that employs, among others, FEA techniques for modeling and analyzing complex problems arising in real-world physical systems. Besides theory and experimentation, CSE is a key driver for engineering research and has gained increasing importance, thanks to the steadily improving performance of modern computer hardware.

Typical CSE workflows initially select an appropriate model for the underlying physical process. This usually leads to a BVP posed over some physical domain Ω. As a next step, the process seeks a mathematical description of the boundary contours $\partial\Omega$ in the form of NURBS-based [PT12] parametric curves / surfaces or a (sufficiently-dense) ordered set of points. Then, a mesh is generated which becomes the geometrical input for approximating the exact BVP-solution using FEA techniques.

Often, the boundary contours $\partial\Omega^{\boldsymbol{\alpha}}$ are a function of a tuple of *shape parameters* $\boldsymbol{\alpha}$, taken from some feasible design space $\boldsymbol{\lambda} \subset \mathbb{R}^k$. The space $\boldsymbol{\lambda}$ may then, for instance, be comprised of all $\boldsymbol{\alpha}$ that parameterize the admissible contours $\partial\Omega^{\boldsymbol{\alpha}}$ corresponding to the design of some engineering device.

Hence, real-world applications require incorporating the steps of $\partial\Omega^{\boldsymbol{\alpha}} \rightarrow \mathcal{T}^{\boldsymbol{\alpha}} \rightarrow$ **FEA** into a fully-automated computational workflow. This is challenging both mathematically and algorithmically because it requires a codebase capable of autonomous intervention in case numerical artifacts occur.

Mesh generation is widely regarded as the weakest link in this chain. The reason for this is two-fold: Firstly, numerical artifacts in $\mathcal{T}^{\boldsymbol{\alpha}}$ manifest themselves in the form of cell degeneracy which rules out the possibility to perform FEA completely. Secondly, unlike in the FEA step, artifacts can typically not be mitigated or removed through refinement. Hence, the algorithmically convenient possibility of trading increased computational resources in exchange for improved robustness may not be given.

A further challenge lies within aspects of computational differentiability: in light of enabling cost-efficient computational design, it is desirable to combine the $\partial\Omega^{\boldsymbol{\alpha}} \rightarrow$ **FEA** pipeline with gradient- or even Hessian-based optimization routines. Hence, the chain $\boldsymbol{\alpha} \rightarrow \partial\Omega^{\boldsymbol{\alpha}} \rightarrow \mathcal{T}^{\boldsymbol{\alpha}} \rightarrow$ **FEA** needs to be differentiated with respect to the shape parameters $\boldsymbol{\alpha}$. Unfortunately, this can be challenging or even impossible, due to the often complex or inaccessible dependency between $\partial\Omega^{\boldsymbol{\alpha}}$ and $\mathcal{T}^{\boldsymbol{\alpha}}$ and the convoluted $\boldsymbol{\alpha}$-dependency it introduces in the FEA residual. Hence, algorithms that admit deriving a closed-form expression of the differential of $\mathcal{T}^{\boldsymbol{\alpha}}$ simplify this step. However, in practice, employing such algorithms alone can be restrictive. As such, they are often combined with (colored) finite difference approximations [GMP05] or automatic differentiation [GWX17], adding further complexity to the already extensive codebase.

Isogeometric Analysis (IGA) is a numerical technique that was conceived in an effort to streamline the typical geometry-to-FEA workflow. This is accomplished by integrating the NURBS-based geometric modeling tools that are characteristic for CAGD into FEA.

In IGA, a NURBS-based mapping operator is directly used for a pullback of the BVP posed on Ω into the parametric domain $\hat{\Omega}$, where the same NURBS-space is employed as a basis for standard FEA techniques. This is referred to as the *isoparametric principle*.
By this, IGA completely bypasses the requirement for traditional mesh generation, potentially circumventing the aforementioned automation challenges. However, the surface-to-volume problem $\partial\Omega^{\boldsymbol{\alpha}} \rightarrow \Omega^{\boldsymbol{\alpha}}$ becomes the IGA-analogue to classical mesh generation. Besides potentially reducing the data conversion overhead resulting from employing tools from both CAGD and FEA in tandem, the geometry is furthermore represented exactly because the step $\partial\Omega^{\boldsymbol{\alpha}} \rightarrow \partial\Omega_h^{\boldsymbol{\alpha}}$ is avoided (see Figure 1.3).

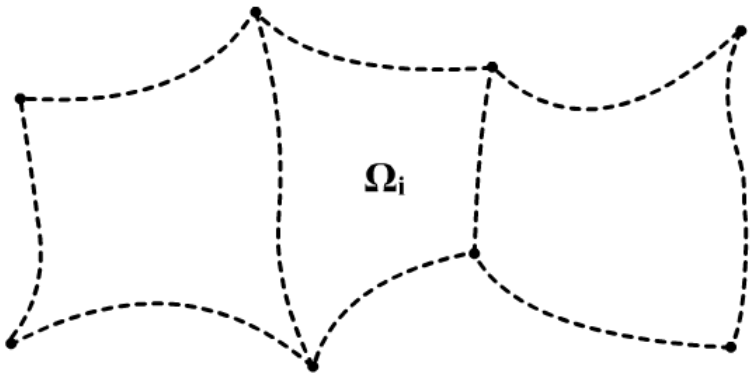


Figure 1.3: Instead of finding a tesselation $\mathcal{T}$, such that $\Omega_h = \text{Int}(\bigcup_{K_i \in \mathcal{T}} \overline{K}_i)$, IGA subdivides Ω into a (relatively) small number of subdomains $\Omega_i \subset \Omega$ which are all homeomorphic to the unit quadrilateral. BVPs posed over Ω are then approximately solved by performing a sequence of pullbacks from the Ω_i into the reference domain using the NURBS-based mapping operators $\mathbf{x}_i : (0,1)^n \rightarrow \Omega_i$. Hereby, the boundary of Ω is represented exactly because the NURBS-based boundary correspondence $\partial\hat{\Omega} \rightarrow \Omega$ is not discretized (cf. Figure 1.1).

1.3. MOTIVATION AND OBJECTIVES

In an effort to bridge the gap between FEA and CAGD, IGA employs the geometric modeling tools associated with CAGD as a basis for analysis using finite element techniques. So far, attempts to consolidate both fields are showing promising results [DVBRS11] but have been largely unidirectional.
Since CAGD algorithms for the surface-to-volume problem have traditionally focused on yielding FEA-compatible tesselated approximations Ω_h of Ω, the pipeline $\partial\Omega \rightarrow \partial\Omega_h \rightarrow \Omega_h$ has received the majority of research interest. Hence, algorithms that operate in the order $\partial\Omega \rightarrow \Omega \rightarrow \Omega_h$, which could be applied in an IGA-setting by skipping the last step, are under-represented. A notable exception is *transfinite interpolation* [GH73, GT82], which, as the name suggests, parameterizes Ω by blending the (in $\mathbb{R}^2$ typically four) segments of $\partial\Omega$ into the interior. While this is a cheap and differential operation, degeneracy of the resulting parameterization $\mathbf{x} : \hat{\Omega} \rightarrow \Omega$ is common. Furthermore, it does not straightforwardly generalize to domains that are not topologically equivalent to the unit quadrilateral in $\mathbb{R}^n$.
Motivated by a bidirectional consolidation of CAGD and FEA, the objective of this dis-

sertation is to extend the available geometry parameterization arsenal with robust algorithms that address the CAGD problem $\partial\Omega \rightarrow \Omega$ with FEA techniques. We seek techniques that are capable of parameterizing two- and *two-and-a-half*-dimensional domains in $\mathbb{R}^n$, $n \in \{2,3\}$ which are topologically equivalent to some convex $\hat{\Omega} \subset \mathbb{R}^m$, with $m \leq n$. With the typical CSE workflow (see Section 1.2) in mind, this work is based on the following main principles:

A. Robustness is more important than efficiency;

B. Differentiability counts;

C. Whenever possible, topology changes should be avoided.

Point A allows for sacrificing a degree of parametric quality in exchange for automation and code simplicity. It favors a framework that can *refine away* mapping degeneracies rather than attempting to repair them through additional *if-else* clauses within the codebase.
Point B implies that a parameterization algorithm should always be designed with differentiability in mind in order to avoid or minimize the requirement to fall back on finite differencing or automatic differentiation.
Point C is of importance when more than one unit quadrilateral (a so-called *patch*) is required to parameterize Ω (hence making $\hat{\Omega}$ a *multipatch* domain). It enables employing a constant number of patches when the boundary contours change as a smooth function of, for instance, time.
In time-dependent settings, points C and B enable extrapolating functions on $\Omega(t)$ from time-instant t_i to t_{i+1} in the static parametric domain rather than having to perform a computationally expensive projection from $\Omega(t_i)$ to $\Omega(t_{i+1})$. Furthermore, regarding time as an additional spatial dimension enables parameterizing 2.5-dimensional solids by sweeping a (nonconstant) parametric surface in t-direction.
To conform with principles A to C, this work presents techniques that seek a mapping $\mathbf{x} : \hat{\Omega} \rightarrow \Omega$ whose inverse is comprised of harmonic functions in Ω. This can be formulated as a PDE-problem, which is then approximately solved using FEA / IGA techniques in $\hat{\Omega}$. Hereby, avoiding topology changes is greatly facilitated by the role of patches in IGA versus elements in FEA. While the number of vertices and the degrees of freedom (DOFs) associated with a single FEA-element are typically of the same order, the DOFs associated with NURBS-bases defined on a single patch typically exceed the number of patch vertices by several orders of magnitude. As such, a patch can be regarded as a *macro element* which enables capturing important geometrical features using a flexible NURBS-based mapping operator rather than increasing the local number of elements which, inadvertently, changes the topology. As a result, IGA typically employs fewer rather than more (macro) elements.

Remark. While the need to change the required number of macro elements (patches) is reduced, the number of DOFs per patch (which are associated with the IGA-analogue of classical elements) may vary in each iteration. However, the absence of (macro) element changes enables defining restriction and prolongation operators in the static parametric rather than the physical domain.

In order to validate their usability in computational workflows, this work's goal is to furthermore apply the proposed techniques for the fully automated parameterization of twin-screw machine [SSK05, ME88] type geometries in two and three spatial dimensions. Hereby, we aim to exploit many of the potential numerical advantages that result from principles A to C. Finally, we aim to combine these principles into a fully differentiable gradient-based shape optimization algorithm and present first results.

1.4. DISSERTATION OUTLINE

The chapters of this work are based on publications. They can be divided into two broad categories: Academic and applied. The academic chapters develop the methodology which is then validated for use within real-world engineering workflows in the applied chapters. This work validates the proposed techniques by applying them to geometries of twin-screw machine type. In the following, we give a color-coded chapter outline whereby the color indicates whether a chapter should be considered academic (blue), applied (red) or mixed (yellow).

Chapter 2: Elliptic Grid Generation Techniques in the Framework of Isogeometric Analysis Applications
This chapter introduces the general concept of PDE-based parameterization and presents first applications.

Chapter 3: An IGA Framework for PDE-Based Planar Parameterization on Convex Multipatch Domains
This chapter extends the methodology from Chapter 2 to multipatch domains.

Chapter 4: Goal-Oriented Adaptive THB-Spline Schemes for PDE-Based Planar Parameterization
This chapter extends the PDE-based methodology with a posteriori strategies aimed at autonomous intervention in case degeneracies are detected. For this, an unstructured spline technology is adopted. Furthermore, the framework is extended with techniques to control the parametric properties of the resulting mapping.

Chapter 5: The Role of PDE-Based Parameterization Techniques in Gradient-Based IGA Shape Optimization Applications
This chapter introduces a gradient-based shape optimization algorithm in which the geometry parameterization becomes part of the problem formulation rather than a preceding step, by adding it in the form of an additional PDE-constraint. It adopts most of the techniques introduced in the preceding chapters and computes the gradient using an adjoint formulation.

Chapter 6: Spline-Based Parameterization Techniques for Twin-Screw Machine Geometries
This chapter introduces a fully automated parameterization framework for twin-screw machine geometries which is largely based on the techniques from Chapter 2.

Chapter 7: Boundary-Conforming Finite Element Methods for Twin-Screw Extruders using Spline-Based Parameterization Techniques
This chapter further develops the framework from Chapter 6 and applies it to twin-screw machine extruder geometries. For this, the techniques from Chapter 3 are adopted.

The chapters employ the same basic notation. However, it is slightly varied depending on the category (academic vs. applied) and the current requirements. By default, we employ the abuse of notation

$$(\mathcal{V}_h)^n = \underbrace{\mathcal{V}_h \times \cdots \times \mathcal{V}_h}_{n \text{ terms}}. \tag{1.1}$$

For better readability, we avoid the parenthesis when no confusion is possible, i.e., $(\mathcal{V}_h)^2 = \mathcal{V}_h^2$. The same is true for tensorial bases, i.e., $(\mathcal{V}_h)^{2\times 2} = \mathcal{V}_h^{2\times 2}$.
Whenever necessary, the chapters recapitulate the current notation in a separate section entitled *Chapter Notation*.

REFERENCES

[BRI05] Anders Bondeson, Thomas Rylander, and Pär Ingelström. *Computational electromagnetics*, volume 51. Springer Science & Business Media, 2005.

[Dic79] B Dickinson. Efficient solutions of linear equations with banded toeplitz matrices. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(4):421–423, 1979.

[DVBRS11] L Beirao Da Veiga, Annalisa Buffa, Judith Rivas, and Giancarlo Sangalli. Some estimates for h–p–k-refinement in isogeometric analysis. *Numerische Mathematik*, 118(2):271–305, 2011.

[GH73] William J Gordon and Charles A Hall. Transfinite element methods: blending-function interpolation over arbitrary curved element domains. *Numerische Mathematik*, 21(2):109–129, 1973.

[GMP05] Assefaw Hadish Gebremedhin, Fredrik Manne, and Alex Pothen. What color is your jacobian? graph coloring for computing derivatives. *SIAM review*, 47(4):629–705, 2005.

[GR12] Vivette Girault and Pierre-Arnaud Raviart. *Finite element methods for Navier-Stokes equations: theory and algorithms*, volume 5. Springer Science & Business Media, 2012.

[GT82] William J Gordon and Linda C Thiel. Transfinite mappings and their application to grid generation. *Applied Mathematics and Computation*, 10:171–233, 1982.

[GWX17] Yisheng Gao, Yizhao Wu, and Jian Xia. Automatic differentiation based discrete adjoint method for aerodynamic design optimization on unstructured meshes. *Chinese Journal of Aeronautics*, 30(2):611–627, 2017.

[Knu01] Patrick M Knupp. Algebraic mesh quality metrics. *SIAM journal on scientific computing*, 23(1):193–218, 2001.

[LMTS96] Roland W Lewis, Ken Morgan, HR Thomas, and Kankanhalli N Seetharamu. *The finite element method in heat transfer analysis*. John Wiley & Sons, 1996.

[ME88] HEH Meijer and PHM Elemans. The modeling of continuous mixers. part i: The corotating twin-screw extruder. *Polymer Engineering & Science*, 28(5):275–290, 1988.

[PT12] Les Piegl and Wayne Tiller. *The NURBS book*. Springer Science & Business Media, 2012.

[Red13] B Dayanand Reddy. *Introductory functional analysis: with applications to boundary value problems and finite elements*, volume 27. Springer Science & Business Media, 2013.

[SSK05] Nikola Stosic, Ian Smith, and Ahmed Kovacevic. *Screw compressors: mathematical modelling and performance calculation*. Springer Science & Business Media, 2005.

[War79] Edward Waring. Vii. problems concerning interpolations. *Philosophical transactions of the royal society of London*, (69):59–67, 1779.

[Yan86] TY Yang. *Finite element structural analysis*, volume 2. Prentice-Hall Englewood Cliffs, NJ, 1986.

[ZTNZ77] Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, Perumal Nithiarasu, and JZ Zhu. *The finite element method*, volume 3. McGraw-hill London, 1977.

2

ELLIPTIC GRID GENERATION TECHNIQUES IN THE FRAMEWORK OF ISOGEOMETRIC ANALYSIS APPLICATIONS

This chapter is based on the publication from [HMV18]. It develops the basic principles of PDE-based parameterization within the context of *Isogeometric Analysis* (IGA) applications and provides a gentle mathematical introduction to many of its core concepts. Besides computational aspects, practical topics such as choosing / adjusting the parametric properties of the boundary contours are discussed. The methods developed in this chapter constitute the main ingredients of the twin-screw machine geometry parameterization frameworks from Chapters 6 and 7.

Generating an analysis-suitable computational grid from a description of no more than the boundaries of the geometry Ω is a frequently occurring problem in numerical analysis. Most classical meshing techniques for finite volume, difference or finite element applications, such as the *Advancing Front Method* [Sch97], *Delaunay Triangulation* [She96] and elliptic or hyperbolic meshing schemes [TSW98], operate with straight-sided elements for generating structured and unstructured computational meshes. Generating high quality meshes from curved elements is still considered a challenging task. A recent development is the introduction of *Isogeometric Analysis* (IGA) [HCB05], which employs NURBS [PT12], the de facto standard in computer aided design (CAD) and computer aided geometric design (CAGD), both for the representation of the geometry Ω and as a basis for the finite element analysis on Ω. A mathematical description of the geometry Ω is accomplished via an operator $\mathbf{x}_h : \hat{\Omega} \rightarrow \Omega_h$ that maps the unit quadrilateral in $\mathbb{R}^n$ (typically

with $n \in \{2,3\}$) onto an approximation Ω_h of Ω, utilizing a linear combination of higher-order spline functions. The numerical computations are then transferred from Ω to $\hat{\Omega}$ by performing a *pullback*. Hereby, splines may allow for an accurate description of Ω with fewer elements, potentially reducing the computational effort associated with this step. Furthermore, an analytical description of the geometry can be turned back into a traditional (structured or unstructured) grid by performing a large number of function evaluations in $\mathbf{x}_h$. This can, for instance, be utilized for local refinement without the need for remeshing.
On the other hand, employing curved instead of linear elements requires more sophisticated parameterization techniques. For instance, verifying that the resulting mapping is folding-free is a more involved process than in the classical case.
For the purpose of creating folding-free mappings utilizing spline functions, we present an algorithm that adopts the principles of *Elliptic Grid Generation* (EGG), a PDE-based parameterization technique from the rich literature of classical grid generation. The basic principles of EGG are adapted in order to be compatible with the principles of IGA. In $\mathbb{R}^2$, EGG has particularly appealing properties since the mapping $\mathbf{x} : \hat{\Omega} \to \Omega$, which follows from the exact solution of the governing PDE-problem, is a bijection [Cas91]. Hence, an approach that seeks an analysis-suitable mapping by computing a sufficiently accurate approximation $\mathbf{x}_h$ of $\mathbf{x}$, is plausible.
This chapter presents an algorithm that is capable of generating valid mappings from a large number of geometry contours, including complicated geometries as they arise in industrial applications. This is accomplished by combining EGG with automated reparameterization techniques and a sophisticated numerical approach for solving the governing (nonlinear) equations. The algorithm is equipped with the means to verify the bijectivity of the resulting mapping and with automated defect correction methods in case bijectivity is violated.
We present strategies for generating folding-free mappings for solids resulting from swept surfaces by combining the planar EGG approach with interpolation in the third spatial component. All applications are provided with example geometries.

2.1. INTRODUCTION

A mathematical description of the target geometry Ω forms an integral part of any application within numerical analysis. Unfortunately, most applications only provide a description of $\partial\Omega$ which necessitates solving a surface to volume problem $\partial\Omega \to \Omega_h$ (with $\Omega_h \simeq \Omega$) before finite element analysis (FEA) techniques can be applied. This preceding step tends to contribute substantially to the overall computational costs and often constitutes a robustness bottleneck in the steps $\partial\Omega \to \partial\Omega_h \to \Omega_h \to$ FEA. This is further exacerbated by the usage of linear (i.e., straight-sided) elements which may necessitate a high element density along the boundary to achieve an accurate approximation $\partial\Omega_h$ of $\partial\Omega$.
Elliptic grid generation (EGG) is a popular PDE-based method for generating structured meshes for complex geometries from no more than a description of their boundaries. Traditionally, the parametric description of $\partial\Omega$, which is typically provided by the CAD-pipeline, is turned into a sufficiently dense point cloud which then serves as the Dirichlet

data of the PDE-problem whose solution is approximated utilizing a finite difference approach. Due to the structured nature of this approach, the total amount of mesh vertices is of the same order as the product of the amount of points used in each coordinate direction. Whenever an accurate representation $\partial\Omega_h$ of $\partial\Omega$ is desired, this can lead to a disproportionately large number of vertices, potentially making the approach (and subsequent numerical simulation) unfeasible. This is further exacerbated by the nonlinear nature of the governing equation, which require an iterative approach.
Isogeometric Analysis (IGA) [HCB05] is a recent development in the field of numerical analysis that attempts to bridge the gap between CAD and FEA. Instead of approximating the geometry by a tessellation, the parametric description of $\partial\Omega$ is directly used to build a NURBS-based mapping operator for Ω. This attempt to synergize CAD and FEA comes with the potential of addressing many of the aforementioned feasibility concerns.
In this chapter, we present an algorithm that employs the basic principles of EGG in an IGA-setting. We give a short motivation in Section 2.2, while Sections 2.4 to 2.8 develop techniques aimed at addressing the potential feasibility concerns. In Section 2.10, we propose numerous possible applications within the realm of IGA as well as classical FEA and demonstrate the applicability of the approach in a number of test cases.
It is assumed that the reader is familiar with B-spline functions and hierarchical spline bases [HCB05, Vuo12, FŠJ15].

2.2. MOTIVATION

Solving the surface to volume (or curve to surface in $\mathbb{R}^2$) problem $\partial\Omega \to \Omega_h$ (with $\Omega_h \simeq \Omega$) remains a challenging task, despite increased efforts to address it within the IGA-community in recent years. To the best of our knowledge, there exists no golden standard and different types of geometries often require specially-tailored parameterization methods. In the planar case, the parameterization of a geometry Ω is built from a bivariate tensor-product B-spline space with known boundary control points. The objective is then to choose the inner control points such that the resulting mapping is (i) bijective and (ii) of high parametric quality as measured by some grid quality functional. Consequently, most parameterization methods are based on minimizing a parametric quality cost function. Let $\hat{\Omega} = (0,1)^2$ and let $\mathbf{x} = (x(\xi,\eta), y(\xi,\eta))^T$ be a mapping that satisfies $\mathbf{x}|_{\partial\hat{\Omega}} = \partial\Omega_h$. Furthermore, let

$$J = \begin{bmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{bmatrix} \tag{2.1}$$

and

$$G = J^T J = \begin{bmatrix} \mathbf{x}_\xi \cdot \mathbf{x}_\xi & \mathbf{x}_\xi \cdot \mathbf{x}_\eta \\ \mathbf{x}_\xi \cdot \mathbf{x}_\eta & \mathbf{x}_\eta \cdot \mathbf{x}_\eta \end{bmatrix} \equiv \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \tag{2.2}$$

be the Jacobian and the metric tensor associated with the mapping, respectively.
In the following we list the most common quality functionals [Win81, XMDG13, GENN12].

$$\begin{aligned}
\text{(Area) orthogonality: } & \int_{\hat{\Omega}} g_{12}^2 \mathrm{d}\boldsymbol{\xi} \quad \text{or} \quad \int_{\hat{\Omega}} g_{11} g_{22} \mathrm{d}\boldsymbol{\xi} \\
\text{Liao: } & \int_{\hat{\Omega}} g_{11}^2 + g_{22}^2 + 2 g_{12}^2 \mathrm{d}\boldsymbol{\xi} \\
\text{Winslow: } & \int_{\hat{\Omega}} \frac{g_{11} + g_{22}}{\det J} \mathrm{d}\boldsymbol{\xi} \\
\text{Uniformity: } & \int_{\hat{\Omega}} \|\mathbf{x}_{\xi\xi}\|^2 + \|\mathbf{x}_{\eta\eta}\|^2 + \|\mathbf{x}_{\xi\eta}\|^2 \mathrm{d}\boldsymbol{\xi} \\
\text{Harmonic energy: } & \int_{\hat{\Omega}} \|\mathscr{L}(\mathbf{x})\|^2 \mathrm{d}\boldsymbol{\xi},
\end{aligned} \tag{2.3}$$

where

$$\mathscr{L}(\mathbf{x}) = g_{22}\mathbf{x}_{\xi\xi} - 2g_{12}\mathbf{x}_{\xi\eta} + g_{11}\mathbf{x}_{\eta\eta}. \tag{2.4}$$

Most algorithms minimize either a single or a linear combination of the above cost functions over the unknown inner control points [XMDG13, FŠJ15].
Unfortunately, performing unconstrained minimization does not guarantee bijectivity of $\mathbf{x}$ (see Figure 2.1). An exception to this is the Winslow functional which, as a downside, needs to be initialized with an already bijective mapping (due to the presence of $\det J$ in the denominator). To overcome this shortcoming, constrained minimization

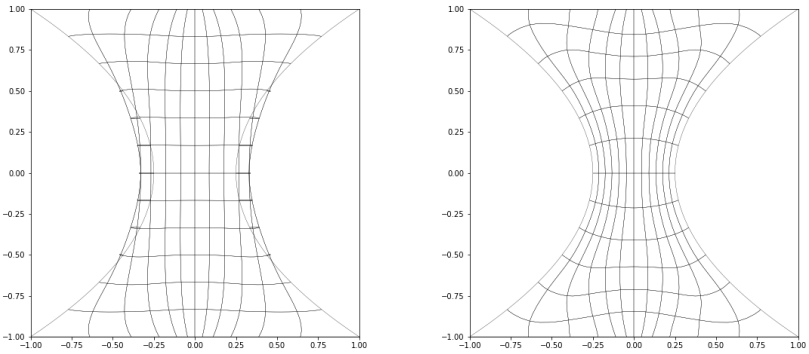


Figure 2.1: A geometry parameterization based on the Liao-function in the unconstrained case (left) and the constrained (right). Unlike in the constrained case, the mapping fails to be bijective in the unconstrained case.

methods are proposed in [GENN12], [XMDG11] and [XMDG10]. They replace the condition $\det J > 0$ in $\hat{\Omega}$ by simpler linear or nonlinear constraints that constitute sufficient conditions for bijectivity. A major challenge, however, is finding an initial guess that is feasible with respect to the chosen constraint.

Our main interest lies in the fast on-the-fly generation of analysis-suitable parameterizations within real-world modelling, simulation and optimization workflows for engineering applications, and hence, we focus not only on generating analysis-suitable parameterizations of high numerical quality but also on aspects of computational efficiency and robustness. This applies both to the efficiency of the process of generating the parameterizations themselves and to their suitability for enabling efficient simulations on heterogeneous high-performance computing (HPC) platforms. In practice, these range from a single workstation to large-scale clusters with hundreds or even thousands of compute nodes.
Hence, our parameterization strategy tries to avoid topology changes, so that an optimized mapping between the collection of patches and available hardware resources, that takes into account the physical proximity of devices and the speed of their interconnects, can be pre-calculated at the beginning of the simulation. To fully exploit the large computing power of modern multi- and many-core hardware architectures at the patch level, the workload per patch needs to be sufficiently high, which motivates our striving for fewer patches with ideally hundreds of thousands of degrees of freedom rather than many patches with only a few hundred.
Our parameterization algorithm is moreover designed with reliability and robustness in mind. That is, it is supposed to produce mappings that are both analysis-suitable and of sufficiently high numerical quality for a wide range of planar geometries without the need for human interaction and/or manual quality post-checks. As mentioned above, we aim at employing fewer rather than more patches. For the test cases considered in this chapter, the optimal topology is usually self-evident. Contrary to methods relying on the quality functionals from (2.3), our method approximately solves a PDE problem based on the principles of Elliptic Grid Generation (EGG).
In the following we present a brief introduction to EGG.

2.3. ELLIPTIC GRID GENERATION

Elliptic grid generation (EGG) addresses the frequently occurring problem of generating an analysis-suitable description of the geometry from a description of only its boundaries. The technique is commonly applied in finite difference and finite volume settings for the generation of structured grids.
Assuming the geometry $\Omega \subset \mathbb{R}^2$ is an open, simply connected domain that is topologically equivalent to the unit quadrilateral $\hat{\Omega} = (0,1)\times(0,1)$, there exists a bijective mapping $\mathbf{x} : \hat{\Omega} \to \Omega$ with inverse $\mathbf{x}^{-1} : \Omega \to \hat{\Omega}$ provided a homeomorphic boundary correspondence $\mathbf{x}|_{\partial\hat{\Omega}} = \partial\hat{\Omega}$ is available. Usually, the mathematical operator $\mathbf{x}$ is not unique, which is why, besides computing a valid (i.e. bijective) mapping, EGG attempts to furthermore satisfy favorable numerical properties, such as a large degree of parametric smoothness.
As elliptic problems are known to yield smooth solutions, EGG imposes the Laplace equation on the components of the inverse-mapping $\mathbf{x}^{-1}$. Assuming that the free topological variables are given by the tuple $\mathbf{x}^{-1} = \boldsymbol{\xi} = (\xi, \eta)^T$, the equation takes the form

$$\Delta\boldsymbol{\xi}(\mathbf{x}) = \mathbf{0} \quad \text{in} \quad \Omega \quad \text{s.t.} \quad \mathbf{x}^{-1}|_{\partial\Omega} = \partial\hat{\Omega}. \tag{2.5}$$

Since one is generally not interested in $\mathbf{x}^{-1}$, the above problem is inverted and scaled in order to yield an equation for $\mathbf{x}$ that is suitable for a computational approach. The

resulting equations read

$$\mathscr{L}(\mathbf{x}) = \mathbf{0} \quad \text{s.t.} \quad \mathbf{x}|_{\partial\hat{\Omega}} = \partial\Omega, \tag{2.6}$$

with the functional $\mathscr{L}(\cdot)$ as defined in (2.4).
The rationale behind imposing the Laplace equation on $\mathbf{x}^{-1}$ (as opposed to $\mathbf{x}$) follows from the observation that the mapping inverse maps into a convex domain. As a result, the solution of (2.6) is bijective [Cas91, Chapter 9].
Furthermore, it can be shown [GENN12] that the solutions of (2.5) and (2.6) are equal to the unique minimizer of the *Winslow functional* from (2.3). Also, (2.6) justifies using the *harmonic energy* functional from (2.3). As the exact solution of (2.6) is bijective, algorithms based on the principles of EGG belong to the minority of approaches that can reliably produce bijective mappings without the need for constraining. To the best of our knowledge, the only other planar parameterization method with this property is proposed in [NC16].
Generally, equation (2.6) cannot be solved analytically. In the following, we present a numerical approach which is based on the principles of IGA.

2.4. DISCRETIZATION

Traditionally, the system of equations (2.6) is approximately solved with a finite difference approach, yielding a discrete set of grid points with known connectivity (see [TSW98]). This approach makes sense whenever a structured representation of the geometry with linear elements is desired as in most classical settings.
In an IGA-setting, the most natural choice of tackling (2.6) is a Galerkin approach over some pre-chosen spline space $\mathcal{V}_h \subset H^2(\hat{\Omega})$. This space can either result from a tensor-product B-spline basis or a (truncated) hierarchical spline basis [GJS12], where the former has the advantage of being structured and the latter has the advantage of allowing for local refinement, potentially converging to a bijective mapping $\mathbf{x}$ with fewer degrees of freedom (see Section 2.8). The approximation of $\mathbf{x}$ is then of the form

$$\mathbf{x}_h(\xi,\eta) = \sum_{k=1}^{N} \mathbf{z}_k w_k(\xi,\eta), \tag{2.7}$$

where the basis $\{w_1,\dots,w_N\}$ spans $\mathcal{V}_h$ and the $\mathbf{z}_k \in \mathbb{R}^2$ denote the corresponding control points. We denote the spline-based mapping by $\mathbf{x}_h$ as opposed to $\mathbf{x}$ in order to stress that it should be regarded as an approximation of $\mathbf{x}$. The approximate nature of $\mathbf{x}_h$, apart from the truncation error introduced by the scheme, results from the condition $\mathbf{x}|_{\partial\hat{\Omega}} = \partial\Omega$, which may have to be discretized, too (see Section 2.5).
The discretized weak counterpart of (2.6) leads to

$$\begin{aligned} &\text{find } \mathbf{x}_h \in \mathcal{V}_h^2 \text{ such that} \\ &\int_{\hat{\Omega}} \boldsymbol{\sigma}_h \cdot \mathscr{L}(\mathbf{x}_h)\,\mathrm{d}\boldsymbol{\xi} = 0, \quad \forall \boldsymbol{\sigma}_h \in (\mathcal{V}_h^\circ)^2 \quad \text{and} \quad \mathbf{x}_h|_{\partial\hat{\Omega}} = \partial\Omega_h, \end{aligned} \tag{2.8}$$

with $\partial\Omega_h \simeq \partial\Omega$ and $\mathcal{V}_h^\circ \equiv \mathcal{V}_h \cap H_0^1(\hat{\Omega})$. Here, $H_0^1(\hat{\Omega})$ denotes the subset of all functions with zero trace in $H^1(\hat{\Omega})$. With this approach, any solution of (2.8) will automatically be in the

form of (2.7), as is mandatory in an IGA-setting.
Decomposing $\{w_1, \dots, w_N\}$ into zero and nonzero trace functions in $H_0^1(\hat{\Omega})$ enables defining the index-sets of *inner* and *boundary* control points $\mathcal{I}_I = \{i \in \mathbb{N} \mid w_i \in H_0^1\}$ and $\mathcal{I}_B = \{1, \dots, N\} \setminus \mathcal{I}_I$, respectively. The sum from (2.7) is split into two terms. They correspond to the unknown (inner) and known (boundary) control points, i.e.,

$$\mathbf{x}_h = \underbrace{\sum_{i \in \mathcal{I}_I} \mathbf{z}_i w_i}_{\text{unknown}} + \underbrace{\sum_{j \in \mathcal{I}_B} \mathbf{z}_j w_j}_{\text{known}}, \tag{2.9}$$

where, the $\mathbf{z}_j$ follow from the discretization of the Dirichlet data (see Section 2.5). With (2.9) in mind, we associate the nonlinear residual $\mathbf{F}(\cdot)$ with (2.8), whereby our objective is to determine its root, i.e.,

$$\mathbf{F}(\mathbf{c}) = \mathbf{0}. \tag{2.10}$$

Here $\mathbf{c} \in \mathbb{R}^{2N_0}$ is a concatenation of the $\mathbf{z}_i$ from (2.9), while N_0 denotes the dimension of $\mathcal{V}_h^\circ|$.
The requirement $\mathcal{V}_h \subset H^2(\hat{\Omega})$ results from the presence of second order derivatives in (2.8). In IGA, choosing a compatible basis is straightforward (unlike in a classical FEA setting). To allow for (locally) reduced (i.e., $C^0(\hat{\Omega})$) regularity, we perform partial integration on the weak form of (2.5). This leads to

$$\begin{aligned} &\text{find } \mathbf{x}_h \in \mathcal{V}_h^2 \quad \text{such that} \\ &\int_{\Omega_h} \frac{\partial \boldsymbol{\sigma}_h}{\partial \mathbf{x}_h} : \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}_h} \mathrm{d}\boldsymbol{x} = 0, \quad \forall \boldsymbol{\sigma}_h \in (\mathcal{V}_h^\circ)^2 \quad \text{and} \quad \mathbf{x}_h|_{\partial\hat{\Omega}} = \partial\Omega_h, \end{aligned} \tag{2.11}$$

where $A : B$ denotes the Frobenius inner product between matrices A and B.
Here, the integral has been carried out over Ω_h as opposed to $\hat{\Omega}$ in order to enable partial integration. Upon pullback into $\hat{\Omega}$, the integral quantity from (2.11) takes the form

$$\int_{\Omega_h} \frac{\partial \boldsymbol{\sigma}_h}{\partial \mathbf{x}_h} : \frac{\partial \boldsymbol{\xi}}{\partial \mathbf{x}_h} \mathrm{d}\boldsymbol{x} = \int_{\hat{\Omega}} \frac{1}{\det J(\mathbf{x}_h)} (\dots) \, \mathrm{d}\boldsymbol{\xi}. \tag{2.12}$$

Like (2.8), (2.11) leads to a nonlinear root-finding problem in $\mathbf{c}$. Hence, an iterative solution approach is mandatory, requiring an initial guess.
In the case of system (2.8), any initial guess $\mathbf{c}^0$ may be substituted into the residual. In (2.12), however, the Jacobian determinant $\det J(\mathbf{x}_h)$ appears in the denominator. Hence, the initial guess must satisfy

$$\forall (\xi, \eta)^T \in \hat{\Omega} : \quad \det J(\mathbf{x}_h)(\xi, \eta) > 0, \tag{2.13}$$

in order to avoid division by zero.
This implies that it already needs to be a bijection itself. Clearly, this defeats the purpose of the approach. However, system (2.12) can be utilized to improve the parametric properties of an already bijective mapping that was, for instance, produced by (2.8). A better

option in this case is directly minimizing the Winslow functional:

$$m_w = \frac{g_{11} + g_{22}}{\det J}. \tag{2.14}$$

The minimization problem takes the form

$$\int_{\hat{\Omega}} m_w \mathrm{d}\boldsymbol{\xi} \to \min_{\mathbf{x}_h \in \mathcal{V}_h^2}, \quad \text{s.t.} \quad \mathbf{x}_h|_{\partial\hat{\Omega}} = \partial\Omega_h, \tag{2.15}$$

which can typically be performed without constraining since $(\det J)^{-1}$ naturally penalizes (nearly) unfeasible solutions. It is seen that (2.15), like (2.12), is compatible with locally reduced regularity but requires a bijective initial guess. However, it is superior to (2.12) since it seeks the global minimizer of (2.15) over $\mathcal{V}_h^2$, which may not coincide with the root of (2.12).
Hence, starting with a basis of reduced regularity is not a viable choice in practice. Instead, Section 2.9 proposes ways to acquire a mapping with reduced regularity from a mapping that is initially $C^{\geq 1}(\hat{\Omega})$-continuous.

2.5. CONTOUR APPROXIMATION AND THE CHOICE OF BASIS

Section 2.3 assumes that a parametric description $\mathbf{x}|_{\partial\hat{\Omega}}$ of $\partial\Omega$ is given. However, $\mathbf{x}|_{\partial\hat{\Omega}}$ may be incompatible with $\mathcal{V}_h \setminus \mathcal{V}_h^\circ$. Denoting the images of southern, eastern, northern and western sides of $\partial\Omega$ by Γ_1, Γ_2, Γ_3 and Γ_4, respectively, we assume that the corresponding boundary transformations, $\mathbf{f}_i : \bar{\gamma}_i \to \bar{\Gamma}_i$, with

$$\bigcup_i \bar{\Gamma}_i = \partial\Omega \quad \text{and} \quad \bigcup_i \bar{\gamma}_i = \partial\hat{\Omega} \tag{2.16}$$

parameterize a Jordan curve in $\mathbb{R}^2$.
The $\mathbf{f}_i(s)$ come in the form of analytic functions or splines curves. Here, we also consider the case in which $\mathbf{x}|_{\partial\hat{\Omega}}$ is only discretely available as a collection of points $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_M\} \subset \mathbb{R}^2$ (as is common in engineering applications).
In the following, we present techniques for choosing spline spaces capable of resolving the contours to a user-defined accuracy. The resulting fits correspond to the Dirichlet data in (2.9).
Sections 2.5.1 to 2.5.3 consider tensor-product B-spline spaces. The generalization to hierarchical splines is sketched in Section 2.5.4.

2.5.1. ANALYTIC CONTOURS

For an analytic function input, we approximate $\partial\Omega$ using an $L_2(\partial\hat{\Omega})$-projection. Given the initially uniform knot vectors Ξ_ξ and Ξ_η with corresponding bivariate $p \geq 2$ (where p denotes the polynomial degree) B-spline space $\mathcal{V}_{h,0} \subset C^1(\hat{\Omega})$, we project the components of $\mathbf{f}_i$ onto $\mathcal{V}_{h,0} \setminus \mathcal{V}_{h,0}^\circ$. To avoid mismatches, the corner control points are constrained to the function evaluations of the $\mathbf{f}_i$ at the end points.
Let $\mathbf{f}_i^\circ$ be the constrained least-squares fit of $\mathbf{f}_i$. We introduce the following element-wise

average residual function:

$$E(\mathbf{f}_i^\circ) = \sum_{\epsilon_j} \frac{1}{|\epsilon_j|} \int_{\epsilon_j} \left\| \mathbf{f}_i^\circ(\xi) - \mathbf{f}_i(\xi) \right\|^2 \mathrm{d}\xi, \tag{2.17}$$

where the ϵ_j denote the one-dimensional elements on $\partial\hat{\Omega}$ (which follow from the unique values of the knot vectors) and $|\epsilon_j|$ their lengths in the parametric domain. The residual from (2.17) measures the quality of the fit both globally and locally. We assess the element-wise quality using the contribution of each summand to the overall sum, serving as a local refinement criterion. An element is marked for refinement whenever the average residual magnitude exceeds some threshold value δ. Here, an element is refined by adding a new knot in the center. We repeat the steps of **project → dyadically refine marked elements** until all local contributions are deemed sufficiently small.
Once the algorithm terminates, a sequence of nested spline spaces $\mathcal{V}_{h,i}, i \in \{0,\dots,m\}$ with corresponding boundary control points $\{\mathbf{z}_j^i\}$ and segmentations of $\partial\hat{\Omega}$ into disjoint elements is available. We employ the resulting hierarchy of refinements to construct robust initial guesses for a computational approach of (2.8) in Section 2.6.3.

2.5.2. B-SPLINE CONTOURS

When the contours are splines themselves, we may choose the basis spanning $\mathcal{V}_h$ as the tensor-product of the bases associated with the input spline curves. As the knot vectors corresponding to the sides (Γ_1,Γ_3) and (Γ_2,Γ_4) may differ, we take the union of the knot vector pairs associated with each coordinate direction. This can lead to a basis of large cardinality, potentially making further computations less feasible. Here, the principles from Section 2.5.1 may be a remedy. We employ the same approach to compute an approximation of the given contours to any desired accuracy, the only difference being that a quadrature scheme needs to be defined with respect to the knot vector union corresponding to the $\mathcal{V}_{h,i}$ and the input spline curve. This ensures that the functions involved are C^∞-continuous on the unified knot spans.

Remark. Only the right hand side vector of the linear system associated with the least-squares fit needs to be assembled in this way. The mass matrix can still be assembled over the knot spans associated with $\mathcal{V}_{h,i}$.

As the input contours may result from a dense knot vector, this step can be computationally expensive. However, considering that the projection corresponds to a univariate integral, the computational costs are still negligible compared to the expected costs of solving equation (2.8). With this approach, we were often able to reduce the basis cardinality by a factor of four or more with insignificant loss of quality.

2.5.3. POINT CLOUD CONTOURS

Many engineering applications provide no more than an ordered set of points as input. Often the point coordinates are a function of, for instance, a set of shape parameters $\boldsymbol{\alpha} \in \boldsymbol{\lambda}$ that tune the design of an engineering device.
When this is the case, we split the input point cloud into four parts

$$P^i = \{\mathbf{p}_1^i, \mathbf{p}_2^i, \dots, \mathbf{p}_{M_i}^i\} \subset \mathbb{R}^2, \quad i \in \{1,2,3,4\}, \tag{2.18}$$

each being assigned to one side of $\partial\hat{\Omega}$. As before, we assume that the first and last points of the P^i correspond to the convex corners of the domain. The next step is constructing a spline fit. The fit can either be exact or approximate. Selecting any monotonically increasing sequence $\{\xi_1 = 0, \xi_2, \ldots, \xi_{M_i-1}, \xi_{M_i} = 1\}$ of parametric values, the objective is constructing four spline curves $\mathbf{f}_i$ for which

$$\left\| \mathbf{f}_i(\xi_j) - \mathbf{p}_j^i \right\|, \quad j \in \{1, \ldots, M_i\}$$

is either zero (exact) or below a threshold (approximate). This is accomplished by recursively constructing the $\mathbf{f}_i$ from spline bases of successively increasing cardinality, reminiscent of Section 2.5.1. As before, the individual contributions to the mismatch serve as a criterion for local dyadic refinement.
Alternatively, we may collocate each point cloud using a FITPACK [Die95] routine and approximate the collocation with the techniques from Section 2.5.2. Again, this yields a hierarchy of refinements with dyadic structure. The dyadic refinement exactly dictates the location of knots. This is advantageous when taking the union of two or more differing knot vectors to, for instance, prolong several mappings to a unified grid. The dyadic structure tends to reduce the knot density compared to an approach with arbitrarily placed knots.
In practice, a collocation greatly benefits the robustness. Since the least-squares matrix associated with a direct fit may be (nearly) rank-deficient, an underdetermined problem is avoided. However, since it requires an exact fit through the (possibly dense) point sets P^i, higher computational costs can be expected.

Remark. A stabilized direct regression is proposed in Section 6.5.

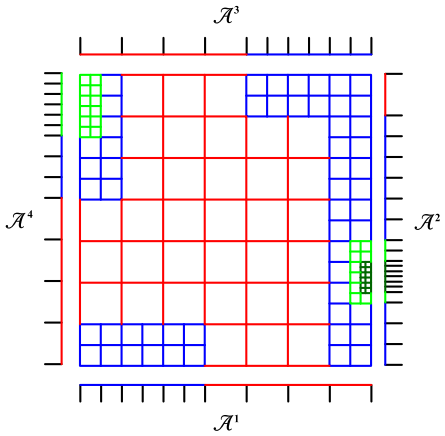


Figure 2.2: A partition of $\hat{\Omega}$ into elements follows from hierarchical refinements of the $\gamma_i \subset \partial\hat{\Omega}$ into the boundary elements $\mathscr{A}_i, i = 1,2,3,4$. The boundary refinements are extended into the interior in the canonical way shown above. The various colors correspond to the levels in the element hierarchy.

2.5.4. HIERARCHICAL SPLINE SPACES

Equation (2.8) is compatible with (truncated) hierarchical spline spaces. Let

$$\mathscr{B}_h = \{w_1, \dots, w_N\} \subset H^2(\hat{\Omega})$$

be a tensor-product reference spline basis. The analogue of the dyadic knot refinement from Sections 2.5.1 to 2.5.3 selects functions $w_i \in \mathscr{B}_h$ that are nonvanishing on $\partial\hat{\Omega}$ and marks them for refinement whenever the projection residual over their supporting elements exceeds a threshold. They are then replaced by their finer counterparts from the spline hierarchy. This process is depicted in Figure 2.2.

The potential advantage of employing hierarchical spline spaces are the degree of freedom (DOF) savings made possible by local refinement. This enables capturing complicated boundary contours without introducing potentially unwanted DOFs in the interior. On the other hand, predicting which parts of the interior require refinement for properly approximating (2.8) is difficult.

In the remainder of this chapter, we restrict ourselves to tensor-product spline spaces. Appropriate schemes for computing $\mathbf{x}_h$ from a (truncated) hierarchical basis are developed in Chapter 4.

2.6. COMPUTATIONAL APPROACH

After the procedure from Section 2.5 has been completed, we are in the possession of a sequence of domain segementations of mutually disjoint elements

$$\mathscr{H}_i = \{K_1, \dots K_m\}, \quad \text{s.t.} \quad \operatorname{Int}\left(\bigcup_{K_j \in \mathscr{H}_i} \bar{K}_j\right) = \hat{\Omega}, \quad \forall i \in \{0, 1, \dots, m\}, \tag{2.19}$$

with corresponding spline spaces $\mathscr{V}_{h,i}$, $i \in \{0, 1, \dots, m\}$ and steadily improving approximations $\partial\Omega_h$ of $\partial\Omega$ from the $\mathscr{V}_{h,i} \setminus \mathscr{V}^\circ_{h,i}$.

Sections 2.6.1 and 2.6.2 propose computational approaches for solving equation (2.10) while Section 2.6.3, discusses the choice of the initial guesses. The computational approaches have been implemented in the Python-library Nutils [vZvZV+19].

2.6.1. TRUNCATED NEWTON APPROACH

Equations (2.8) or (2.12), discretized over some appropriately chosen spline space $\mathscr{V}_h$, lead to a root-finding problem of the form

$$\mathbf{F}(\mathbf{c}) = 0, \tag{2.20}$$

where $\mathbf{c}$ refers to the internal (unknown) degrees of freedom.

Due to the nonlinear nature of this equation, an iterative approach to finding its root $\mathbf{c}$ is mandatory. Even though algorithms based on the principles of EGG traditionally employ a Picard-based approach (as in [LB07] and [Man89]), we shall present an approach that is based on Newton's method. Defining

$$\mathbf{F}^k \equiv \mathbf{F}\left(\mathbf{c}^k\right) \tag{2.21}$$

and

$$\left[\frac{\partial \mathbf{F}}{\partial \mathbf{c}}\right]^k \equiv \left.\frac{\partial \mathbf{F}}{\partial \mathbf{c}}\right|_{\mathbf{c}=\mathbf{c}^k}, \tag{2.22}$$

a Newton-step is computed as the solution of

$$\left[\frac{\partial \mathbf{F}}{\partial \mathbf{c}}\right]^k \delta \mathbf{c}^k = -\mathbf{F}^k. \tag{2.23}$$

The next iterate becomes

$$\mathbf{c}^{k+1} = \mathbf{c}^k + \kappa \delta \mathbf{c}^k, \tag{2.24}$$

where $\kappa \in (0,1]$ is a damping factor. The Nutils-internal Newton-solver is equipped with sophisticated line-search capabilities. Assuming that

$$\left\| \mathbf{F}\left(\mathbf{c}^k + \kappa \delta \mathbf{c}^k\right) \right\| = A + B\kappa + C\kappa^2 + D\kappa^3, \tag{2.25}$$

the constants A, B, C, D are estimated from the current and updated tangents and residuals and κ is selected such that (2.25) is minimized in the Euclidean norm in order to reduce the required number of residual evaluations.
The above procedure is repeated until

$$\left\| \mathbf{F}\left(\mathbf{c}^n\right) \right\|_2 < \epsilon, \tag{2.26}$$

where ϵ is some convergence threshold.

2.6.2. PSEUDO TIME-STEPPING

Should the Newton-approach fail to converge, the solver can fall back on the technique of pseudo time-stepping [CKK03] which trades a higher chance of convergence for an increased computational effort. In practice, we rarely have to fall back on this approach thanks to the reliability of the truncated Newton-approach in combination with a robust initial guess (see Section 2.6.3).
Pseudo time-stepping seeks the steady-state solution of

$$\frac{\partial}{\partial t}\mathbf{x}_h(t) = -\mathscr{L}(\mathbf{x}_h(t)), \tag{2.27}$$

with $\mathscr{L}(\cdot)$ as defined in (2.4).
We discretize in space utilizing a Galerkin approach and in time utilizing a backward Euler scheme. This leads to:

$$\begin{bmatrix} M & 0 \\ 0 & M \end{bmatrix} \frac{\mathbf{c}^{k+1} - \mathbf{c}^k}{\Delta t_k} = -\mathbf{F}^{k+1}, \tag{2.28}$$

where M denotes the mass matrix corresponding to the canonical basis of $\mathcal{V}_h \cap H_0^1(\hat{\Omega})$.
The drawback of this discretization is its nonlinear right-hand side term. We circumvent this issue with a first order expansion around $\mathbf{F}^k$

$$\mathbf{F}^{k+1} \simeq \mathbf{F}^k + \left[\frac{\partial \mathbf{F}}{\partial \mathbf{c}}\right]^k \delta \mathbf{c}^k, \quad \text{where} \quad \delta \mathbf{c}^k = \mathbf{c}^{k+1} - \mathbf{c}^k. \tag{2.29}$$

Substitution and rearrangement yields

$$\left(\frac{1}{\Delta t_k}\begin{bmatrix} M & 0 \\ 0 & M \end{bmatrix} + \left[\frac{\partial \mathbf{F}}{\partial \mathbf{c}}\right]^k\right)\delta \mathbf{c}^k = -\mathbf{F}^k. \tag{2.30}$$

As proposed in [KK98], the time step selection is based on the following recursive formula

$$\Delta t_k = \Delta t_{k-1}\frac{\left\|\mathbf{F}^{k-1}\right\|_2}{\left\|\mathbf{F}^k\right\|_2}. \tag{2.31}$$

Again, the iteration is terminated once

$$\left\|\mathbf{F}\left(\mathbf{c}^k\right)\right\|_2 < \epsilon. \tag{2.32}$$

2.6.3. CHOOSING AN INITIAL GUESS

The computational approaches from Subsections 2.6.1 as well as 2.6.2 are iterative and thus require an initial guess $\mathbf{c}^0$. The quality of $\mathbf{c}^0$ makes a tremendous difference in the convergence success or failure, as well as the amount of iterations needed. We shall dedicate this section to constructing robust initial guesses.

The preprocessing step from Section 2.5 leaves us with the nested sequence of spline spaces $\mathcal{V}_{h,k}$, $k \in \{0,\dots,m\}$ and corresponding known boundary control points. They each lead to a problem of the form

$$\mathbf{F}_k\left(\mathbf{c}_k\right) = \mathbf{0}, \tag{2.33}$$

where the subscript k in $\mathbf{F}_k$ indicates the level in the hierarchy.

We may utilize this hierarchical structure to our advantage. The idea is to solve $\mathbf{F}_0\left(\mathbf{c}_0\right) = \mathbf{0}$ and manipulate its solution to serve as an initial guess for the problem $\mathbf{F}_m = \mathbf{0}$. An initial guess for $\mathbf{F}_0$ is generated utilizing transfinite-interpolation [Coo67]. It is then solved using the techniques from Section 2.6.1 or 2.6.2. Depending on the complexity of the geometry, the truncated Newton approach typically converges within about $4-6$ iterations. As the dimension of $\mathcal{V}_{h,0}$ is significantly smaller than $\mathcal{V}_{h,m}$, a large number of iterations does not compromise feasibility. After finding the root $\mathbf{c}_0$, we prolong the corresponding mapping from the zeroth level in the hierarchy to $\mathcal{V}_{h,m}$ and impose the discrepancy between the boundary control points (BCPs) of the prolonged coarse-grid solution and the BCPs at the m-th level as a Dirichlet boundary condition on a linear elasticity [Fal08] problem. When the boundary $\partial\Omega$ is complicated, the required number of refinement recursions m is large. To improve the quality of the initial guess, we may then solve some of the intermediate problems in the hierarchy before proceeding to the m-th level. A good guideline is choosing the steps such that the dimension of the problem quadruples after each level.

In the following, we illustrate this approach by applying it to the target contours depicted in Figure 2.3. Here, the left and right subfigures depict the least-squares fit of the contours with respect to $\mathcal{V}_{h,0}$ and $\mathcal{V}_{h,m}$, respectively. Figure 2.4 plots the mapping at the zeroth level in the refinement hierarchy. Here, the $\mathcal{V}_{h,k}$ are spanned by bicubic B-spline

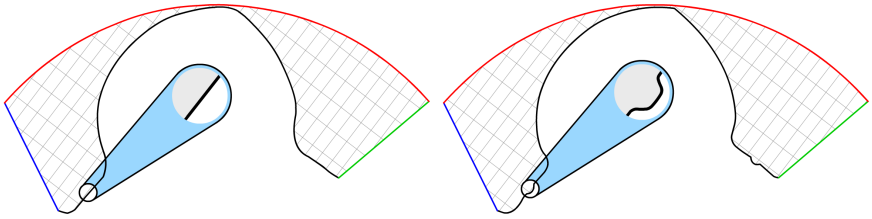


Figure 2.3: Contours corresponding to the first (left) and last level in the hierarchy (right). The figure zooms in onto a small protrusion on the southern boundary that is not properly resolved by the coarse spline fit, while being well-captured using the fine-grid knot vector. The boundaries that correspond to the various sides of the parametric domain $\hat{\Omega}$ have been highlighted in different colors for convenience.

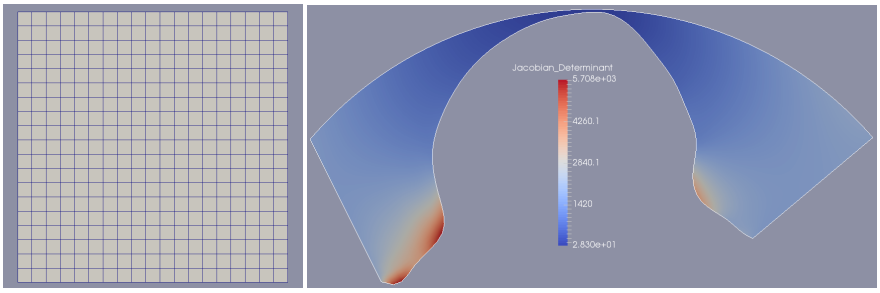


Figure 2.4: The domain segmentation (left) and mapping (right) at the first level in the hierarchy. The color depicts the value the Jacobian determinant assumes at each point.

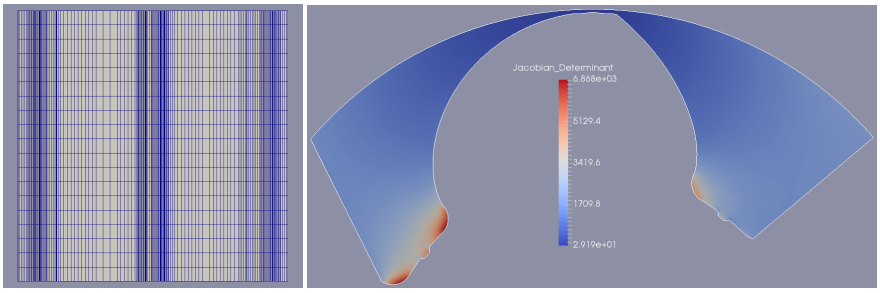


Figure 2.5: Domain segmentation corresponding to the highest level in the hierarchy (left), the initial mapping acquired with the linear elasticity method (right).

bases with maximum regularity. The mapping is prolonged to the canonical B-spline basis of the grid depicted in Figure 2.5 (left). Figure 2.5 (right) shows the mapping generated from the linear elasticity problem, which serves as an initial guess for the m-th level. Plotting the Jacobian determinant reveals that the initial guess succeeds in being bijective and is therefore already analysis-suitable. Solving $\mathbf{F}_m(\mathbf{c}_n) = \mathbf{0}$ thus only potentially improves the parametric quality in this case. Figure 2.6 (right) shows the parametric properties of the initial guess by the narrow gap of the geometry compared to an inferior prolongation method (left).

In practice, the solver typically converges within 3 – 4 iterations on the finest level. We have found this hierarchical approach to greatly improve the robustness as well as the efficiency of the algorithm. When the boundary control points are not the result of one of the methods from Section 2.5 but are fixed from, e.g., the CAD-input, the problem may be tackled with an artificially created hierarchy. This is done by removing knots from the input knot vectors and acting with the Moore-Penrose pseudo inverse [BH12] of the prolongation operator $T : \mathcal{V}_{h,0} \rightarrow \mathcal{V}_{h,m}$ on the boundary control points. Heuristically, the

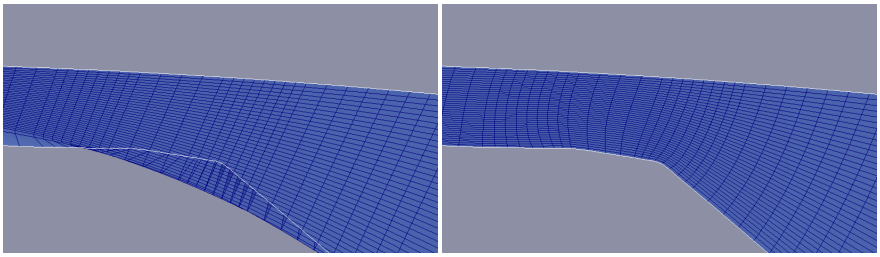


Figure 2.6: The parametric properties by the narrow gap of the geometry upon prolonging the coarse-grid solution to the fine grid using different techniques. The first is acquired by acting with the canonical prolongation matrix on the internal DOFs and simply replacing the boundary DOFs (left). In the second case, the inner DOFs are shifted as a result of a deformation field, which is the solution of a linear elasticity problem (right). Note that the first method fails to yield a bijection.

initial guess acquired with the linear elasticity approach often succeeds in being bijective. Since this approach only requires solving one negligibly small nonlinear and one linear problem, when successful, it is an inexpensive method for finding an analysis-suitable parameterization. One may also choose to use it for simulation without further optimization. In this case we might regard the linear elasticity mapping as a perturbation of the optimized mapping. This furthermore suggests that linear elasticity may be an inexpensive mesh-update strategy.
Distortions in the mapping have a less severe effect on numerical quality for higher values of the polynomial degree p [LEB+10].

2.7. REPARAMETERIZATION TECHNIQUES

The parametric properties of the contours have a significant influence on the parametric quality of $\mathbf{x}_h$. In the following we discuss various (re-)parameterization techniques and give recommendations about which technique should be used in which setting. When $\partial\Omega$ comes in the form of point clouds P^i, $i \in \{1,\dots,4\}$, reparameterization is accomplished by reassigning monotone increasing parametric values to the $\mathbf{p}^i_j \in P^i$. When the input comes in the form of an analytic function or spline curve, reparameterization requires taking the function composition with a monotone function $g : [0,1] \rightarrow [0,1]$. While this is possible in theory, we have found it to be algorithmically impractical. Hence, in the following we consider a point cloud input. If an analytic or spline input needs to be reparameterized, we first convert it into a dense point cloud by performing a large number of function evaluations.
For arbitrary geometries, an (approximate) arc length parameterization is the default

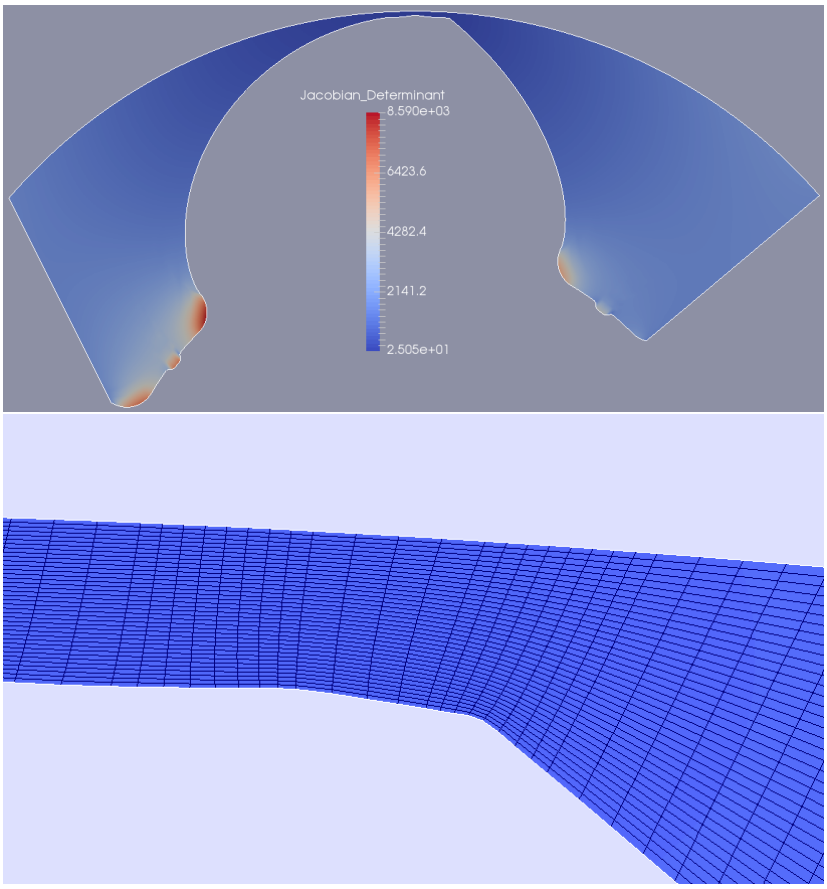


Figure 2.7: The final mapping (left) and its parametric properties by the narrow gap (right).

choice. Given $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_K\}$, this is easily accomplished by properly choosing the collocation abscissae $\{0 = \xi_1, \xi_2, \ldots, \xi_K = 1\}$. Defining l_i recursively by

$$l_i = l_{i-1} + \|\mathbf{p}_i - \mathbf{p}_{i-1}\|, \tag{2.34}$$

starting with $l_1 = 0$, and ending on l_K, we define

$$\hat{\xi}_i = \frac{l_i}{l_K}. \tag{2.35}$$

An approximate arc length (or *chord length*) parameterization corresponds to choosing $\xi_i = \hat{\xi}_i$. In many settings, we have observed that an arc length based parameterization leads to unsatisfactory results.
Figure 2.9 shows the contours of a geometry with challenging characteristics. The challenge arises as a result of the simultaneous presence of very wide and very narrow gaps. As discussed in Section 2.2, a lot of effort has gone into IGA-compatible parameterization

techniques in recent years. However, to the best of our knowledge, only a small number of publications deal with the impact of the parametric properties of the boundary contours on the final mapping. In [XMGR14], Xu et al. propose a technique that changes the parametric properties of a curve while retaining its shape. This is accomplished by representing the curve in a related NURBS-basis over a different knot vector. The parametric properties can be tuned to some quality functional with one given degree of freedom. On the other hand, [XLM+18] attempts to tackle challenging geometries by segmenting them into a large number of Bézier patches via quadrangulation on the Bézier control net. The parametric properties of each patch are then optimized based on a local quality functional.

In the following, we present a reparameterization technique designed for tube-like shaped geometries subject to extreme aspect ratios as the one in Figure 2.9. Here, ensuring that the same parametric value is assumed by the narrow gaps on opposite boundaries is a crucial aspect for the numerical quality of the mapping (see Figure 2.8).

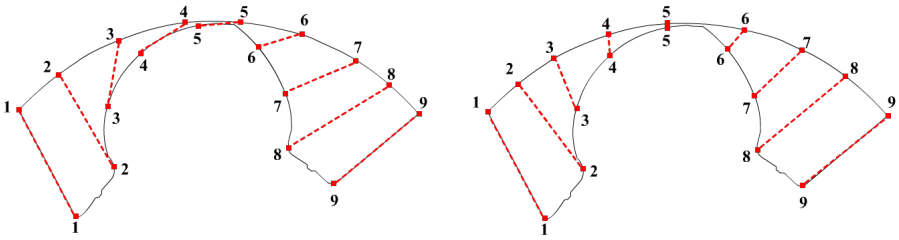


Figure 2.8: Depiction of the expected grid lines for chord length parameterized (left) and constrained chord length parameterized (right) boundary contours. The parametric properties of the boundary contours are of major importance whenever extreme aspect ratios are involved. The chord length parameterization may to lead to unsatisfactory results, whereas the more flexible constrained chord length parameterization produces superior results.

To this end, we have often successfully employed a more flexible variant of the chord length approach, which we shall henceforth refer to as the *constrained chord length parameterization*.

Let the two point clouds $P_1 = \{\mathbf{p}_1^1, \mathbf{p}_2^1, \dots, \mathbf{p}_n^1\}$ and $P_2 = \{\mathbf{p}_1^2, \mathbf{p}_2^2, \dots, \mathbf{p}_m^2\}$ be assigned to the two sides of $\partial\hat{\Omega}$ associated with the same coordinate direction. The constrained chord length selects the parameters $\Xi_2 = \{\xi_1^2 = 0, \xi_2^2, \dots, \xi_m^2 = 1\}$ associated with P_2 based on the values of $\Xi_1 = \{\hat{\xi}_1^1 = 0, \hat{\xi}_1^2, \dots, \hat{\xi}_n^1 = 1\}$, which result from a chord length parameterization (cf. equation (2.35)). Given a threshold distance $\epsilon > 0$ and assuming that a tuple $(\mathbf{p}_i^1, \mathbf{p}_j^2)$ with $\left\| \mathbf{p}_i^1 - \mathbf{p}_j^2 \right\| \leq \epsilon$ exists, let

$$(n_0, m_0) = \underset{(i,j) \in \{1,\dots,n\} \times \{1,\dots,m\}}{\operatorname{argmin}} \left\| \mathbf{p}_i^1 - \mathbf{p}_j^2 \right\|. \tag{2.36}$$

After the tuple (n_0, m_0) has been found, we set $\xi_{m_0}^2 = \xi_{n_0}^1$. We continue in the same fashion on the two subsets of P_1 and P_2 with lower and higher indices as (n_0, m_0), respectively. In practice, we usually remove a few points from the resulting subsets such that (in terms of chord length) two matched tuples of points have a predefined minimum distance with respect to one another. This process is repeated in a hierarchical fashion until

no more points that are eligible for matching are found. The procedure is illustrated in Figure 2.9.

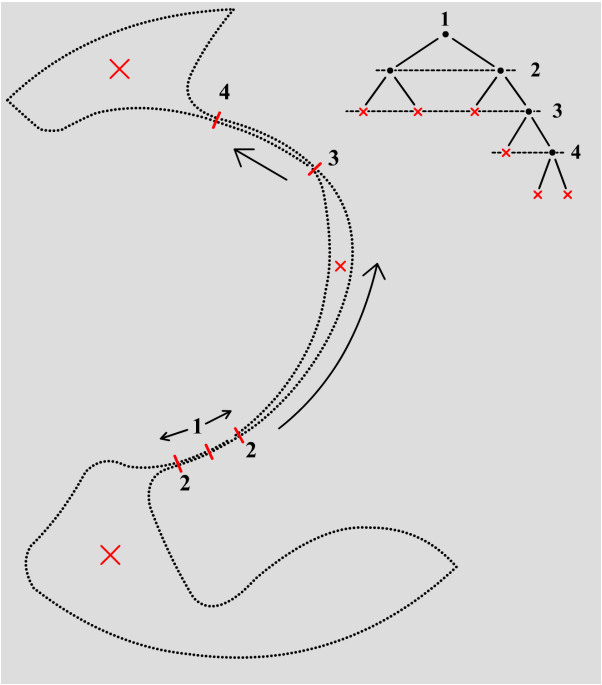


Figure 2.9: The constrained chord length technique seeks the minimum distance between the elements of the two input point clouds, which are then matched. Upon completion, this step is repeated on the point clouds pairs with higher and lower indices and so on. Red crosses indicate that no more point pairs eligible for matching have been found.

After the algorithm terminates, the set Ξ_2 is partially known. It remains to determine the values of the ξ_i^2 with unmatched index. Let $\mathscr{I}_2^{\text{known}} = \{i \in \mathbb{N} \mid \xi_i^2 \text{ is known}\}$ (which includes the first and last point). We assign a value to the remaining ξ_i^2 by carrying out a spline interpolation of $\{\hat{\xi}_2^i \mid i \in \mathscr{I}_2^{\text{known}}\}$ against $\{\xi_2^i \mid i \in \mathscr{I}_2^{\text{known}}\}$. For this, we use a monotone cubic interpolation [FC80] yielding the monotone reparameterization function $g(\xi)$. The remaining parametric values are then given by $\xi_i^2 = g(\hat{\xi}_i^2)$.
Upon completion, the tuples (Ξ_i, P_i) serve as an input for the techniques from Section 2.5. Figure 2.10 depicts parameterizations corresponding to the contour from Figure 2.9 with constrained and unconstrained chord length parameterization, respectively. The plot of the Jacobian determinant reveals that the chord length parameterized geometry fails to be bijective, whereas bijectivity is achieved with the constrained approach. A zoom-in onto the narrow gap shows the properties of the isolines in the constrained and unconstrained case. The lines have to be deemed unsatisfactory in the unconstrained case, whereas the constrained approach yields an outcome that can be considered superior both computationally (bijectivity is achieved) and numerically (the grid lines should are orthogonal to the boundary). On the other hand, the unconstrained mapping is com-

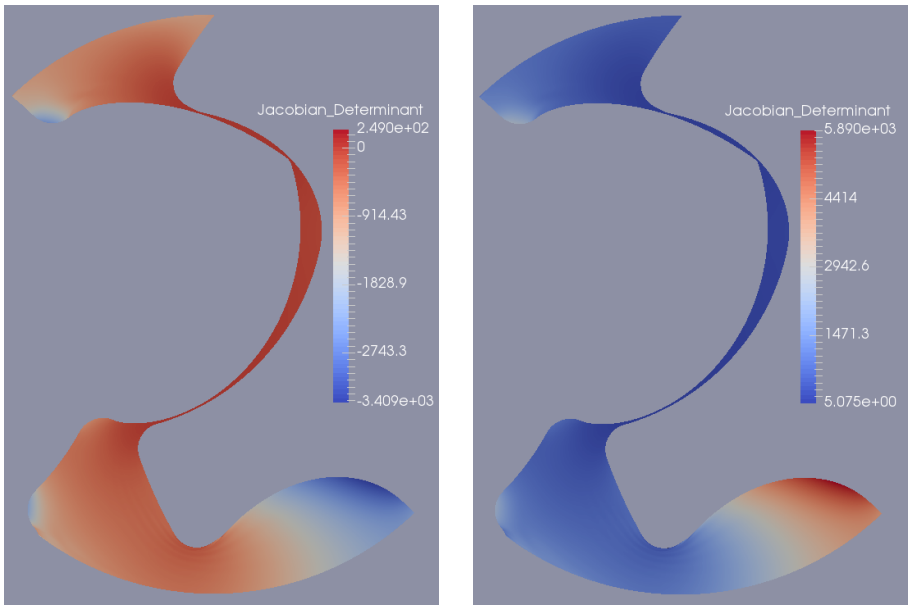


Figure 2.10: The resulting mapping without (left) and with (right) constrained chord length parameterization

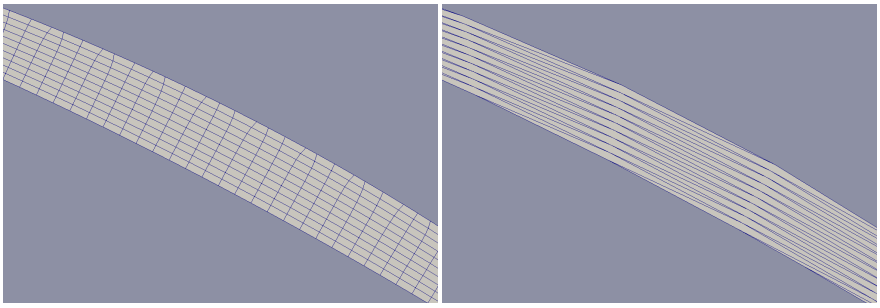


Figure 2.11: A zoom in on the narrow part of the geometry shows its grid lines before and after parameterization.

prised of 2366 DOFs, while the corresponding number is 3953 in the constrained case. This is likely due to the parametric velocity not being (approximately) constant over the parametric interval $[0, 1]$, requiring a locally higher density of basis functions to reach the approximation tolerance. Hence, the resulting computational costs are slightly higher.

Remark. In our experience, a suitable parameterization of $\partial\Omega_h$ is a critical aspect of ensuring the quality of $\mathbf{x}_h$, regardless of the parameterization technique employed. The reparameterization technique discussed in this subsection yields the best results in a wide range of applications. Since it operates on a discrete point-cloud, however, it is an inherently discrete process, which can yield qualitatively different outcomes for similar inputs. The technique is further developed in Chapter 6.

2.8. POST PROCESSING

Upon convergence, the mapping $\mathbf{x}_h(\xi,\eta)$ is post processed. Due to the approximate nature of (2.8), numerical errors can compromise the theoretically predicted bijectivity property of the exact solution. The task of the post processor is detecting defects caused by numerical errors. Defects manifest themselves as sign changes in the Jacobian determinant [Pro15]

$$\det J(\mathbf{x}) = \det\left[\frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}}\right]. \tag{2.37}$$

Assuming that Ω_h is positively oriented, the objective of the post-processor is verifying that $\det J$ is strictly positive.
In practice, this is accomplished by projecting $\det J$ onto a spline space that contains it, as in [GENN12], and requiring that all the projection weights d_i are positive. Thanks to the positivity of spline functions, this is a sufficient condition for bijectivity but not a necessary one. Heuristically, we have observed that $\det J > 0$ implies $d_i > 0$ in many cases. However, in the presence of extreme aspect ratios (as in Figure 2.10), this sufficient condition can be too restrictive. In practice, we may replace the harsh requirement $\det J > 0$ in $\hat{\Omega}$ by $\det J > 0$ on all quadrature points of the numerical integration scheme utilized for the subsequent isogeometric analysis on Ω_h. We fall back onto this requirement whenever the sufficient condition is impractically restrictive.
If the post check finds a defect (a negative Jacobian determinant), we locally refine the basis in a small region centered around the location of the defect and add another level $m+1$ to the hierarchy. The defective mapping is prolonged to the refined basis and passed back to the solver where it serves as an initial guess. This process is repeated until no more defects are detected. We have often successfully corrected defects while freezing the weights corresponding to basis functions supported on elements which are far removed from the location of the defects. By this, fewer DOFs need to be recomputed which reduces the computational costs.

Remark. A more in-depth discussion of refinement strategies is given in Chapter 4.

As an example, we consider the geometry depicted in Figure 2.12. Here, $\mathcal{V}_h$ is the result of a knot vector with 19 elements in both coordinate directions. The figure reveals that a defect is located at the upper part of the narrow gap of the geometry. Zooming in onto this part, Figure 2.13 plots the parameterization before and after defect correction. Figure 2.14 depicts the corresponding refinements in the parametric domain.

2.9. LOWER ORDER REGULARITY

As discussed in Section 2.4, using spline spaces with global $C^0(\hat{\Omega})$-continuity can be computationally impractical (requiring a bijective initial guess). In certain settings, a steep angle in the boundary contour is best modeled as a C^0-continuity resulting from a repeated knot. Here, an option is segmenting the geometry and applying the PDE-based problem formulation patchwise. However, it may not always be evident how to segment. Furthermore, segmentation is not always differentiable, making it impractical when the methodology is applied for, e.g., gradient-based shape optimization. Figure 2.15 will

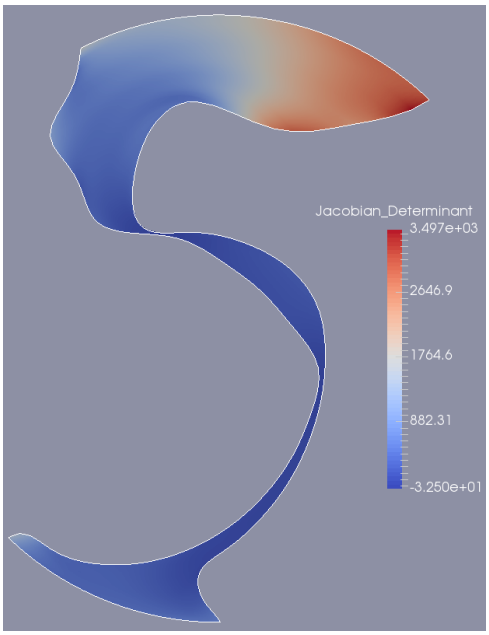


Figure 2.12: The parameterization of a challenging geometry. On first glance the parameterization seems fine, the plot of the Jacobian determinant, however, shows that it assumes negative values.

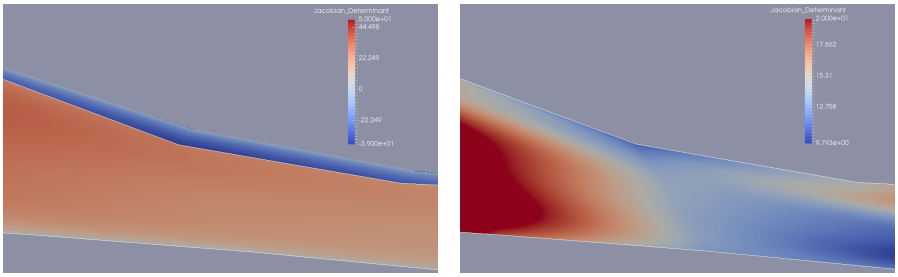


Figure 2.13: Defective part of the geometry (left) and the same part after defect correction (right).

serve as an example geometry for the discussion that follows. The depicted contours $\partial\Omega_h$ of this O-type parameterization are taken from a bicubic spline basis. Knots have been added in the parametric domain to properly capture the C^0-continuities which are marked by red crosses. This leads to a homeomorphic boundary correspondence between the various sides of $\hat{\Omega}$ and Ω_h. However, in order to be compatible with (2.8), the boundary correspondence needs to be diffeomorphic. We lower the knot multiplicity by one wherever it exceeds $p-1$ and restrict the Dirichlet extension to the newly-formed spline space. Henceforth we shall refer to the two spaces formed in this way by $\mathcal{V}_{h,i} \subset C^i(\hat{\Omega})$, $i \in \{1,2\}$. Figure 2.16 shows a zoomed-in picture of the C^0-continuity before and after smoothing. The smoothed contours are passed to (2.8). With Section

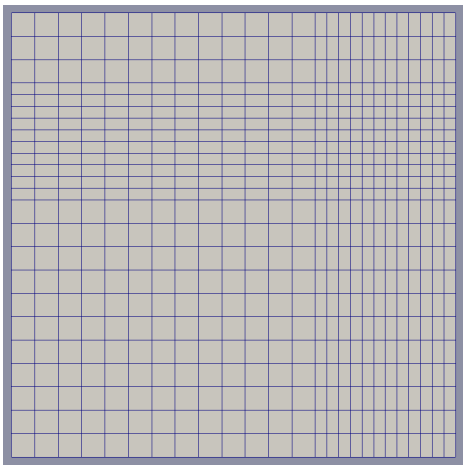


Figure 2.14: The elements that make up $\hat{\Omega}$ after refinement. The post processor has locally performed dyadic refinement of the knot vectors in both coordinate directions.

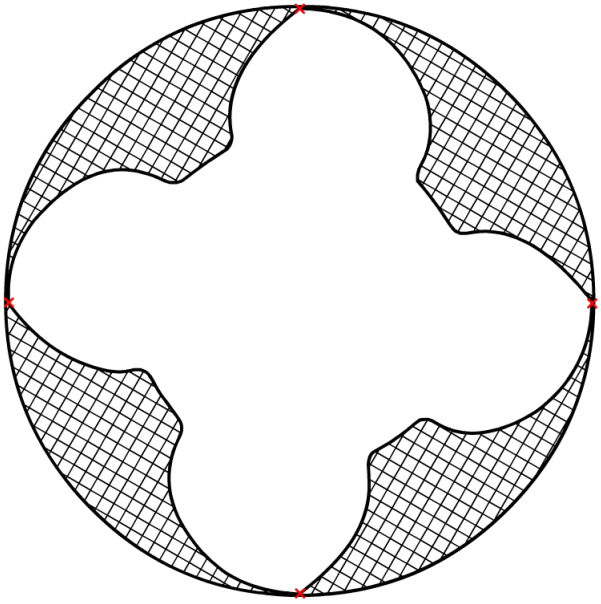


Figure 2.15: Target contours containing C^0-continuities.

2.6.3 in mind, we compute the solution $\mathbf{x}_h \in \mathcal{V}^2_{h,0}$ from its smoothed counterpart using a mesh update strategy. The boundary discrepancy between both mappings is imposed as a Dirichlet boundary condition to a linear elasticity problem. Should bijectivity be retained, the solution is passed to (2.15) as an initial guess to further improve the para-

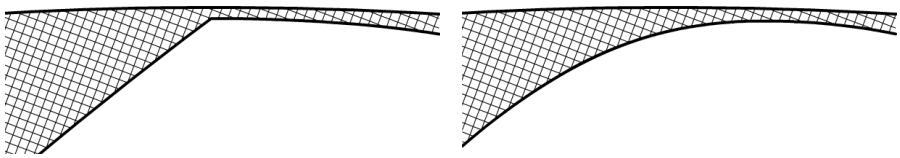


Figure 2.16: Part of the boundary contours before and after smoothing.

metric properties.
Should the linear elasticity solution fail to be bijective, we can perform the above steps with a Dirichlet boundary condition that is dampened by a factor $\alpha < 1$. This process is repeated until the target contours are attained.
Figure 2.17 depicts the geometry that corresponds to the contours from Figure 2.15.

Remark. An approach that supports spline spaces with global $C^0(\hat{\Omega})$ continuity while not requiring a bijective initial guess is proposed in Chapter 3.

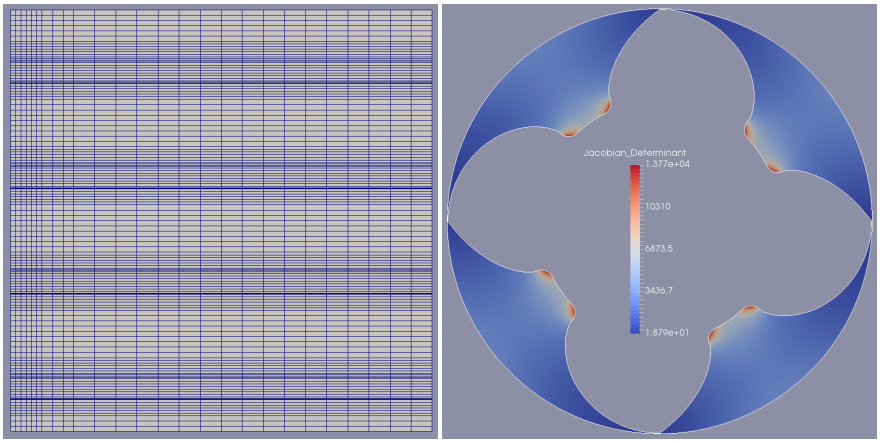


Figure 2.17: The mapping corresponding to the contours from Figure 2.15 with C^0 continuities.

2.10. APPLICATIONS

In this section, we discuss possible applications of the algorithm within both IGA and classical FEA settings.

2.10.1. APPLICATION TO CONSTRAINED OPTIMIZATION

While the proposed methodology produces analysis-suitable parameterizations in a wide range of applications, it lacks the flexibility of exactly controlling the parametric properties of the outcome. In [GENN12], the mapping is optimized by choosing the inner control points based upon the optimization of various quality functionals. As a constraint, the optimizer requires that the projection of $\det J$ onto a spline space which contains

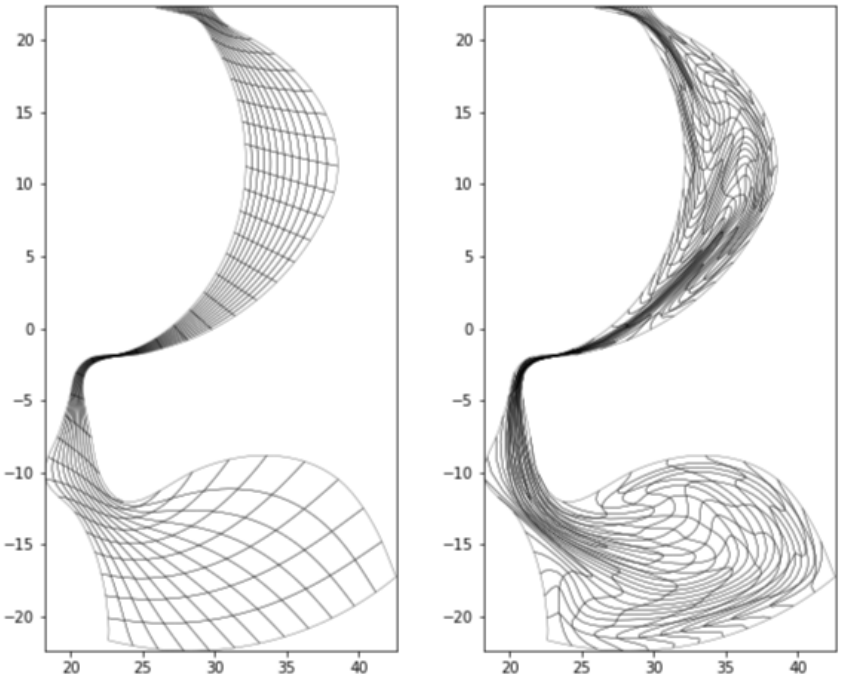


Figure 2.18: The EGG algorithm (left) and the approach from (2.38) (right) applied to the same geometry. Even though both outcomes are feasible with respect to a nonlinear sufficient condition for bijectivity, the EGG mapping is highly regular, while (2.38) yields a distorted outcome. This is not surprising as (2.38) is solely aimed at yielding a feasible initial guess, regardless of parametric quality.

it only carries positive weights. This constitutes a sufficient condition for bijectivity. A major hurdle is finding an initial guess that satisfies this requirement. Let $\mathbf{d}(\mathbf{c})$ be the vector of projection weights. As before, $\mathbf{c}$ contains the inner control points from (2.9). The authors propose computing an initial guess as the result of the following optimization problem:

$$\begin{aligned} &\max_{\mathbf{c}} z, \\ &\text{s.t. } \mathbf{d}(\mathbf{c}) \geq z\mathbf{1}, \end{aligned} \tag{2.38}$$

where $\mathbf{1}$ is a vector of ones. In the following, we refer to (2.38) as the *max Z* problem. If the optimization routine returns a maximizer with $z > 0$, a feasible initial guess has been found. An optimization routine (typically IPOPT [BZ09]) then further optimizes a quality cost function under the constraint $\mathbf{d}(\mathbf{c}) > \mathbf{0}$.
When solving (2.8) yields a bijective outcome, initializing the quality cost function optimization with the EGG-solution is a plausible alternative to optimizing (2.38). Figure 2.18 shows the result of applying EGG and max Z to the same geometry. Both mappings are feasible with respect to the constraint $\mathbf{d}(\mathbf{c}) > 0$.
In the following, we study the quality of both initial guesses as measured by substitut-

ing them into a number of widely-employed cost functions. Hereby, lower values mean that the initial guess is closer to the optimum, which typically means fewer optimization iterations. Table 2.1 shows the results of substituting into the Liao, Winslow and Area-Orthogonality functionals (see Section 2.2). Generally, the EGG mapping yields a lower outcome. A further striking difference is the required number of iterations, which amount to almost 30 in the case of (2.38), while the iterative Newton solver converges in only five for the EGG mapping. We conclude that EGG is a viable alternative to (2.38)

	# Iterations	Liao	Winslow	Area-Orthogonality
EGG	5	1.25×10^7	38.73	5.54×10^5
$\max Z$	27	2.88×10^8	70.25	5.76×10^6

Table 2.1: A table showing the results of substituting both mappings from Figure 2.18 into various parameterization quality cost functionals, along with the required number of iterations for achieving feasibility with respect to the sufficient condition.

for initializing constrained optimization. We have noted the sequence of steps *transfinite interpolation → EGG mapping → cost function optimization* to be both robust and computationally efficient in practice.

2.10.2. TIME-DEPENDENT GEOMETRIES AND SWEPT SURFACES

We are again considering the geometry from Figure 2.17.
Assuming that we perform numerical simulation by discretizing using IGA in the spatial component and finite differences (with fixed time step) in time, part of the computational approach is generating an analysis-suitable mapping for the geometry at the current time instant. For the O-grid, the northern and southern boundaries are disregarded and a periodic knot vector is utilized in the η-direction. If the boundary correspondence is a smooth function of time, previous mappings may be reused for forming better initial guesses at the current time instant through extrapolation. Assuming the grid to be fixated at the eastern (casing) and sliding along the western boundary (rotor), we can construct high-quality initial guesses by extrapolating the mappings from previous time steps to the current time step. Assuming Δt corresponds to some fixed angular increment $\Delta\theta$, after each iteration we act with the canonical rotation matrix on the inner contour-interpolation $\mathbf{c}_r(\eta)$ and reparameterize to let $\eta = 0$ coincide with $\mathbf{c}_r(\eta)_y = 0$. After the first mapping $\mathbf{x}_1(\xi,\eta)$ has been computed utilizing the principles from Section 2.6, the internal DOFs of $\mathbf{x}_1$ serve as an initial guess for computing $\mathbf{x}_2$. This constitutes a zeroth order extrapolation. Once more mappings become available, higher order extrapolations can be constructed. Here, we restrict ourselves to a sequence of length six, i.e., we construct an initial guess for $\mathbf{x}_n$ from $\{\mathbf{x}_{n-6},\dots,\mathbf{x}_{n-1}\}$ or a subset thereof. A sequence of length l is utilized to construct a $(l-1)^{\text{th}}$-order spline extrapolation for each internal DOF which is then evaluated at the current time instant and utilized as an initial guess. This operation is computationally efficient since each DOF can be treated independently. Note also that this operation is compatible with variable time steps.
Figure 2.19 shows a realization of the sliding O-grid algorithm from Section 2.10.2 with $\Delta\theta = 0.003$ after 0 and 125 iterations. In practice, the truncated Newton iteration converges after only one iteration once enough grids for a 5^{th}-order extrapolation are avail-

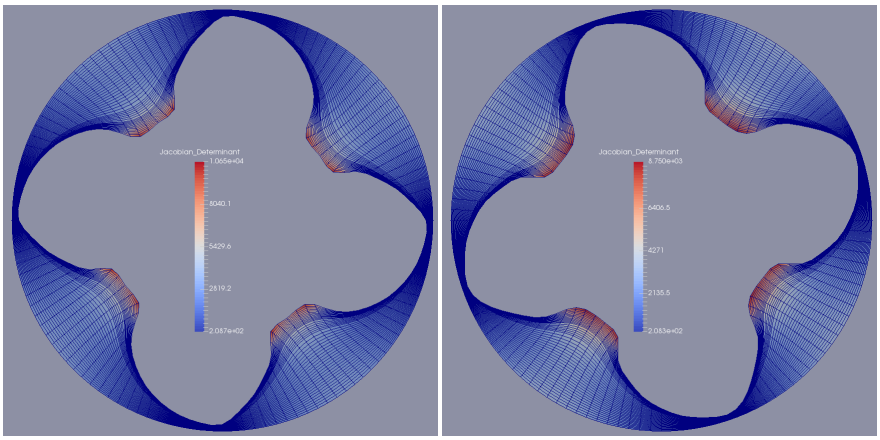


Figure 2.19

able. This is a remarkable result since the computational costs are reduced to the same level as algebraic methods (solving one linear system), while retaining the favorable properties of EGG, such as bijectivity.
Noting that above time stepping process essentially computes planar *slices* of a swept surface (whose boundary is parameterized by a mutually disjoint family of contours $\partial\Omega_h(t)$, $t \in [0,T]$), we replace the temporal axis t by the z-axis and perform first-order interpolation (in z) between the tuples $(\mathbf{x}_i, \mathbf{x}_{i+1})$, $\forall i \in \{1,\dots,125\}$. Figure 2.20 depicts the resulting swept surface geometry. The positivity of the Jacobian determinant reveals that the interpolation, too, is analysis-suitable. The non-smooth nature of the first order interpolation, however, is clearly visible. This methodology is essentially equivalent to performing a collocation on a family of problems $\mathbf{F}_t(\mathbf{c}_t) = \mathbf{0}$, $\forall t \in [0,T]$. Here, we require that $\mathbf{F}_{n\Delta t}(\mathbf{c}_{n\Delta t}) = \mathbf{0}$, where $n \in \mathbb{Z}$ and Δt denotes the spacing. If a $p = 1$ basis resulting from a uniform knot vector with spacing Δt is utilized in t-direction, the problem becomes separable (in t) enabling us to replace a volumetric problem by a sequence of planar problems, as above. Using higher-order bases in t-direction leads to a higher-order interpolation between the slices. However, for $p \geq 2$, the system loses separability. Should the collocation fail to yield a bijection, despite all its constituent slices $\mathbf{x}_i$ being bijective, we refine in t and add more collocation points. Fortunately, we can utilize our database of m planar slices to construct excellent initial guesses for the additional required slices via interpolation.

2.10.3. PARAMETERIZING THE INTERIOR OF A SCREW-MACHINE

Figure 2.21 (left) shows the contours of a screw machine geometry. The objective is generating a parameterization of the interior between the two rotors and the casing represented by two circular arcs (meeting at the so-called CUSP points) utilizing two or more mappings. The figure also shows two smaller circular arcs, that together with parts of the left and right rotors constitute the contours of a geometry, henceforth referred to as the *separator*, whose singlepatch mapping is depicted on the right. We generated the

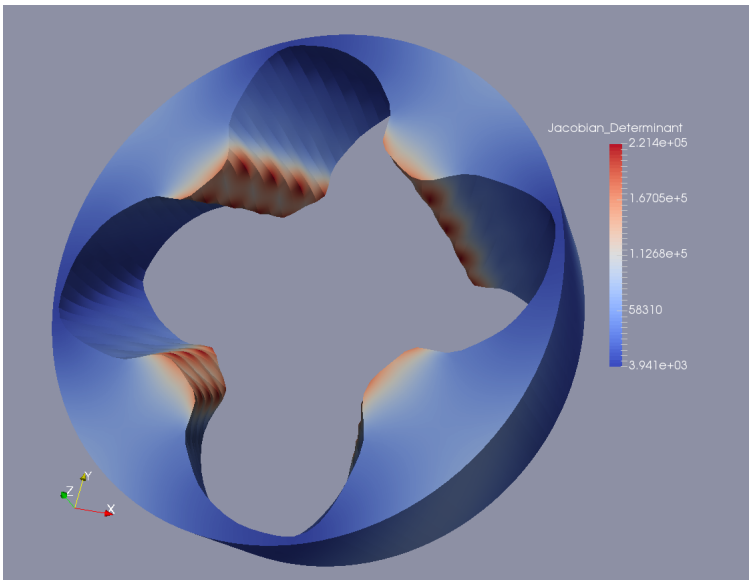


Figure 2.20: First order interpolation of the first 126 cross-sections.

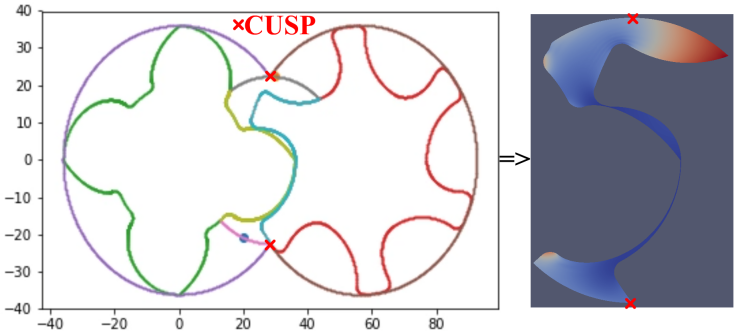


Figure 2.21: Contours of a challenging geometry that can only be parameterized using two or more patches.

mapping using the Newton-approach from Section 2.6.1 in conjunction with the reparameterization technique from Section 2.7. As a next step, we generate a separating line connecting the two CUSP-points of the casing. We select the CUSP-point preimages $\{\mathbf{p}_1, \mathbf{p}_2\} \subset \partial\hat{\Omega}$ and connect them by a curve. The most straightforward choice is a straight line, which already yields decent results. The continuous nature of $\mathbf{x}_h$, however, also allows for a more flexible choice. In this case, we traverse the parametric domain such that the geometry is split most-evenly into two disjoint parts by the narrow gaps of the separator. The procedure is illustrated in Figure 2.22. The splitting curve is exported as a dense point cloud and utilized to split the target geometry into two parts. As a next step, two geometry contours are formed using the splitting curve, parts of the left and the right rotors and connecting line segments (see Figure 2.23). We generate mappings for

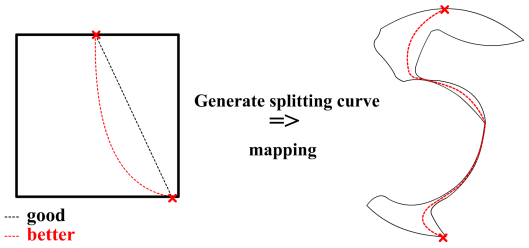


Figure 2.22: Generating a high-quality splitting curve utilizing the bijective mapping.

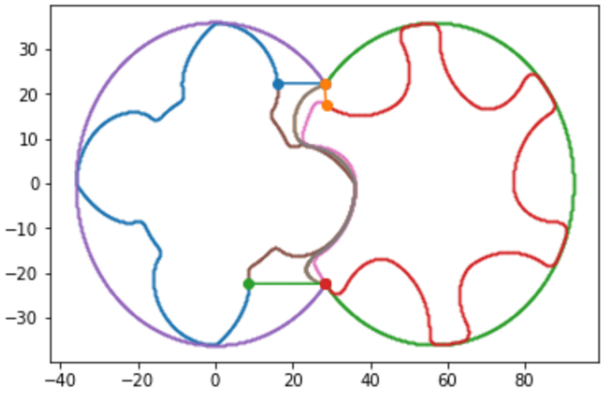


Figure 2.23: Four-patch target geometry.

the geometries, fenced off by these contours, using the same technique employed in the generation of the separator, whereby both geometries utilize the same knot vector in the η-direction. The result is depicted in Figure 2.24 (left). In Figure 2.24 (right) the red line

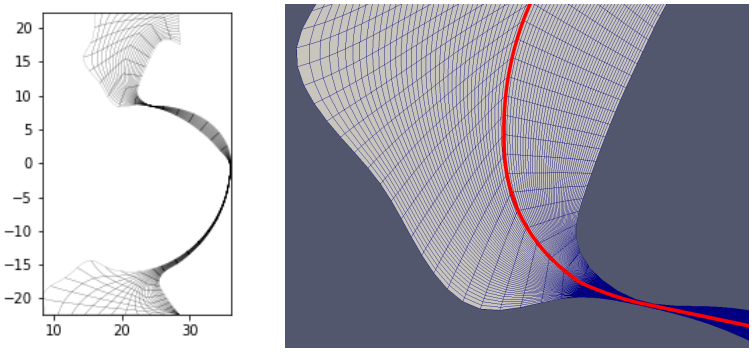


Figure 2.24: Two-patch mapping for the middle part of the geometry depicted in Figure 2.23 (left). The right picture shows the segment at which the two geometries meet. It is clearly seen that the two mappings are conforming but only C^0-continuous across the interface.

shows the segment at which the two mappings meet. Since the mappings are conforming at the interface curve, we merge them through C^0-coupling. Upon completion, the separator is the image of a single-patch mapping with a p-fold repeated internal knot at $\xi = 0.5$, where p denotes the polynomial order. As a next step, we remove the steep angles at the interface curve by finding the root of system (2.12) using the merged two-patch parameterization as an initial guess. A zoom-in on the resulting mapping is shown in Figure 2.25. Finally, two C-type mappings are generated for the parts to the left and

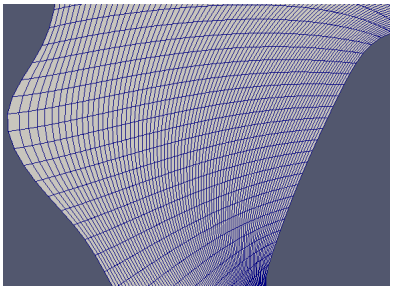


Figure 2.25: Upon combination of the two mappings into one and remeshing, the steep angles are significantly smoothed.

right of the separator. The final geometry, which is the image of a total of three maps, is depicted in Figure 2.26.

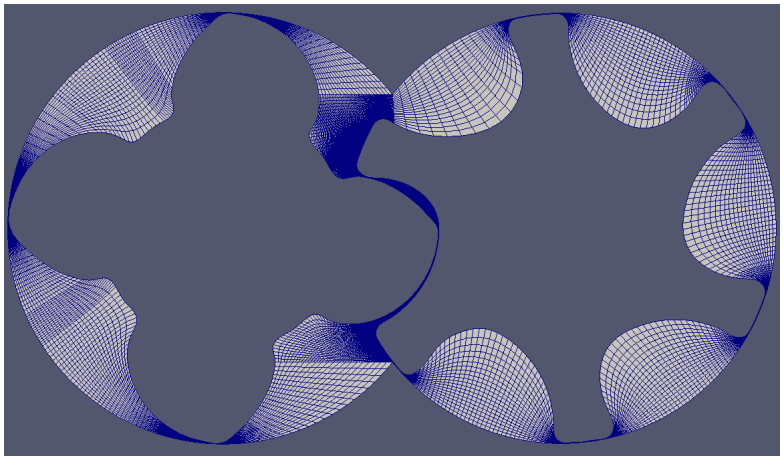


Figure 2.26: The final parameterization of the interior of the screw machine geometry which utilizes a total of three mappings.

2.10.4. APPLICATIONS WITHIN CLASSICAL MESHING

Classical finite difference, volume or element applications typically utilize structured grids comprised of linear elements. Even though the approach proposed in this chapter is based on curved as opposed to linear elements, the mapping operator can be utilized

to generate linear meshes of any desired accuracy.
Given a bijective spline-based mapping $\mathbf{x}_h$, a straight-sided mesh is acquired by performing a large number of function evaluations and connecting the resulting point cloud by linear edges. Thanks to the flexibility of higher-order spline functions, the tolerance of the boundary approximation $\partial\Omega_h$ tends to be attained with fewer elements than in the linear case, which potentially reduces the computational costs. We perform an adequate number of function evaluations in $\mathbf{x}_h$ in order to retain a sufficiently accurate collocation of $\partial\Omega_h$ with linear edges. Hereby, the spacing can be locally tuned to reach the desired accuracy with less elements.
A further application is a tunable grid resolution. Often, it is desirable to have a higher element density, for instance, along the boundary contours in tangential direction. As an example we again consider the familiar geometry from Figure 2.27. We introduce the

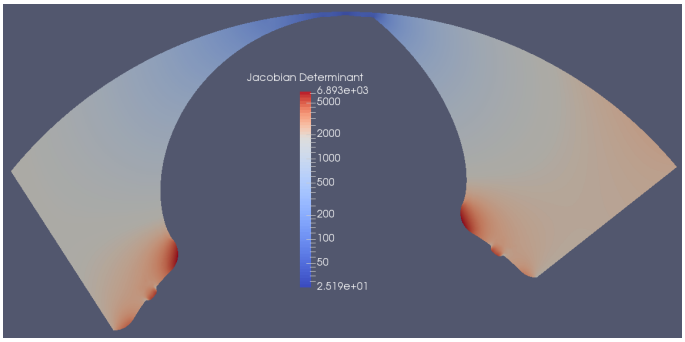


Figure 2.27: The bijective geometry with curved elements.

ordered sets

$$\Xi = \{0, 1/\Delta\xi, 2/\Delta\xi, \dots, 1\},$$

with $\Delta\xi = 1/401$ and

$$\mathscr{H} = \{0, \eta'(\Delta\eta), \eta'(2\Delta\eta), \dots, 1\},$$

with

$$\eta'(\eta) = \frac{1}{2}\left(\frac{\arctan\left(6(\eta - \frac{1}{2})\right)}{\arctan(3)} + 1\right) \tag{2.39}$$

and $\Delta\eta = 1/35$. A straight-sided boundary layer mesh is acquired by evaluating $\mathbf{x}_h$ in the $(\xi, \eta) \in \Xi \times \mathscr{H}$ and connecting neighbouring nodes with edges. The resulting grid is depicted in Figure 2.28. Besides tuning the mesh properties, this methodology allows for generating meshes with vertices whose number exceed the DOFs of the spline mapping by several orders of magnitude. As a single evaluation of $\mathbf{x}_h$ is cheap, generating dense meshes is computationally inexpensive. It should be noted that this process essentially collocates $\mathbf{x}_h$. Hence, bijectivity may be lost if the number of sampling points is

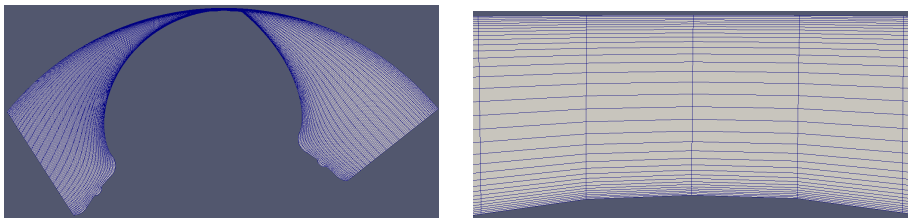


Figure 2.28: A classical mesh acquired from the geometry depicted in Figure 2.27 by performing a large number of function evaluations (left). The right picture is a zoom-in onto the narrow gap showing the boundary element distribution.

too low. Finally, we mention that the mapping can also be utilized to generate unstructured meshes. This is accomplished by performing (for instance) a triangulation in the parametric domain and carrying out function evaluations in the vertices. The value of the Jacobian determinant in the vertices can in turn serve as a local refinement criterion.

2.11. CHAPTER CONCLUSIONS

In this chapter, we presented an algorithm that embeds the principles of EGG into an IGA-framework. This was accomplished by discretizing the weak form of the governing equations with a Galerkin approach using spline basis-functions with global $C^{\geq 1}(\hat{\Omega})$-continuity and solving the resulting nonlinear system with a truncated Newton-approach. Furthermore, we presented automated reparameterization techniques to improve the parametric properties of challenging geometries like, for instance, geometries with extreme aspect ratios.

The hierarchical Newton approach greatly contributes to the overall stability. The algorithm reliably produces analysis-suitable singlepatch parameterizations for a wide range of geometries and typically converges within $3-5$ nonlinear iterations.

Even though self-intersections are uncommon, defects resulting from the truncation error can be located on internal elements. So far, the choice of $\mathcal{V}_h$ is solely based on accurately resolving the boundary contours but does not prevent self-intersections in the interior. Currently, self-intersections are resolved by the defect-correction techniques from Section 2.8.

The *constrained chord length* reparameterization technique introduced in Section 2.7 has been successfully applied to generate parameterizations for tube-like shaped geometries, (see Figure 2.10), as well as the multipatch parameterization from Figure 2.26. Due to its discrete nature, small perturbations in the input data may already yield considerably different reparameterizations. Hence, the technique is only C^{-1}-continuous with respect to the input. This prevents it from being applied to problems in which the parameterization of the boundary contours $\partial\Omega_h(t)$ needs to be homeomorphic in t, such as the sliding grid and swept surface parameterizations from Section 2.10.2. In such cases, computing reparamterizations at discrete time instants $n\Delta t$, $n \in \mathbb{Z}$ and performing a regression over t may be an option. Hereby, it is of major importance that the regression restricted to $t = t_0$ is monotone for each $t_0 \in [0, T]$.

Finally, we proposed a methodology compatible with spline spaces $\mathcal{V}_h \subset H^1(\hat{\Omega})$. This

allows for parameterizing geometries in which the spline-based boundary correspondences between the various sides of $\partial\hat{\Omega}$ and $\partial\Omega$ are only homeomorphic, rather than diffeomorphic. Despite its repeated successful application, the method proposed in Section 2.9 is relatively involved. An approach that allows for spaces $\mathcal{V}_h \subset H^1(\hat{\Omega})$ while not requiring a bijective initial guess constitutes a possible topic for further research.

REFERENCES

[BH12] João Carlos Alves Barata and Mahir Saleh Hussein. The moore–penrose pseudoinverse: A tutorial review of the theory. *Brazilian Journal of Physics*, 42(1-2):146–165, 2012.

[BZ09] Lorenz T Biegler and Victor M Zavala. Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575–582, 2009.

[Cas91] José E Castillo. *Mathematical aspects of numerical grid generation*. SIAM, 1991.

[CKK03] Todd S Coffey, CT Kelley, and David E Keyes. Pseudotransient continuation and differential-algebraic equations. *SIAM Journal on Scientific Computing*, 25(2):553–569, 2003.

[Coo67] Steven A Coons. Surfaces for computer-aided design of space forms. Technical report, DTIC Document, 1967.

[Die95] Paul Dierckx. *Curve and surface fitting with splines*. Oxford University Press, 1995.

[Fal08] Richard Falk. Finite element methods for linear elasticity. 1939, 01 2008.

[FC80] Frederick N Fritsch and Ralph E Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980.

[FŠJ15] Antonella Falini, Jaka Špeh, and Bert Jüttler. Planar domain parameterization with THB-splines. *Computer Aided Geometric Design*, 35:95–108, 2015.

[GENN12] Jens Gravesen, Anton Evgrafov, Dang-Manh Nguyen, and Peter Nørtoft. Planar parametrization in isogeometric analysis. In *International Conference on Mathematical Methods for Curves and Surfaces*, pages 189–212. Springer, 2012.

[GJS12] Carlotta Giannelli, Bert Jüttler, and Hendrik Speleers. THB-splines: The truncated basis for hierarchical splines. *Computer Aided Geometric Design*, 29(7):485–498, 2012.

[HCB05] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *CMAME*, 194:4135–4195, 2005.

[HMV18] J Hinz, M Möller, and C Vuik. Elliptic grid generation techniques in the framework of isogeometric analysis applications. *Computer Aided Geometric Design*, 2018.

[KK98] Carl Timothy Kelley and David E Keyes. Convergence analysis of pseudo-transient continuation. *SIAM Journal on Numerical Analysis*, 35(2):508–523, 1998.

[LB07] P Lamby and KH Brakhage. Elliptic grid generation by B-spline collocation. In *Proceedings of the 10th International Conference on Numerical Grid Generation in Computational Field Simulations, FORTH, Crete, Greece*, 2007.

[LEB+10] Scott Lipton, John A Evans, Yuri Bazilevs, Thomas Elguedj, and Thomas JR Hughes. Robustness of isogeometric structural discretizations under severe mesh distortion. *Computer Methods in Applied Mechanics and Engineering*, 199(5):357–373, 2010.

[Man89] J Manke. A tensor product B-spline method for numerical grid generation. Technical report, Washington Univ., Seattle, WA (USA). Dept. of Applied Mathematics, 1989.

[NC16] Xianshun Nian and Falai Chen. Planar domain parameterization for isogeometric analysis based on teichmüller mapping. *Computer Methods in Applied Mechanics and Engineering*, 311:41–55, 2016.

[Pro15] Marina Prokhorova. Homeomorphism criteria for the theory of grid generation. *arXiv preprint arXiv:1504.01087*, 2015.

[PT12] Les Piegl and Wayne Tiller. *The NURBS book*. Springer Science & Business Media, 2012.

[Sch97] Joachim Schöberl. Netgen an advancing front 2d/3d-mesh generator based on abstract rules. *Computing and visualization in science*, 1(1):41–52, 1997.

[She96] Jonathan Richard Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Applied computational geometry towards geometric engineering*, pages 203–222. Springer, 1996.

[TSW98] Joe F Thompson, Bharat K Soni, and Nigel P Weatherill. *Handbook of grid generation*. CRC press, 1998.

[Vuo12] Anh-Vu Vuong. *Adaptive hierarchical isogeometric finite element methods*. Springer Science & Business Media, 2012.

[vZvZV+19] Gertjan van Zwieten, Joost van Zwieten, Clemens Verhoosel, Eivind Fonn, Timo van Opstal, and Wijnand Hoitinga. Nutils, June 2019.

[Win81] Alan M Winslow. Adaptive-mesh zoning by the equipotential method. Technical report, Lawrence Livermore National Lab., CA (USA), 1981.

[XLM+18] Gang Xu, Ming Li, Bernard Mourrain, Timon Rabczuk, Jinlan Xu, and Stéphane PA Bordas. Constructing IGA-suitable planar parameterization from complex cad boundary by domain partition and global/local optimization. *Computer Methods in Applied Mechanics and Engineering*, 328:175–200, 2018.

[XMDG10] Gang Xu, Bernard Mourrain, Régis Duvigneau, and André Galligo. Optimal analysis-aware parameterization of computational domain in isogeometric analysis. In *International Conference on Geometric Modeling and Processing*, pages 236–254. Springer, 2010.

[XMDG11] Gang Xu, Bernard Mourrain, Régis Duvigneau, and André Galligo. Parameterization of computational domain in isogeometric analysis: methods and comparison. *Computer Methods in Applied Mechanics and Engineering*, 200(23-24):2021–2031, 2011.

[XMDG13] Gang Xu, Bernard Mourrain, Régis Duvigneau, and André Galligo. Constructing analysis-suitable parameterization of computational domain from cad boundary by variational harmonic method. *Journal of Computational Physics*, 252:275–289, 2013.

[XMGR14] Gang Xu, Bernard Mourrain, André Galligo, and Timon Rabczuk. High-quality construction of analysis-suitable trivariate nurbs solids by reparameterization methods. *Computational Mechanics*, 54(5):1303–1313, 2014.

3

AN IGA FRAMEWORK FOR PDE-BASED PLANAR PARAMETERIZATION ON CONVEX MULTIPATCH DOMAINS

This chapter is based on the publication from [HMV19]. The development of this work was motivated by our striving for a practically viable parameterization algorithm that supports spline spaces with reduced regularity (i.e., global $C^{\geq 0}$ continuity). While C^0 bases are compatible with the methodology from Section 2.9, the appearance of the Jacobian determinant in the denominator of the residual makes it impractical for a framework that automatically generates a large number of parameterizations for a wide range of different input contours. This is due to the need to initialize with a nondegenerate mapping (which is typically not available) and numerical issues related to the possibility of dividing by zero, such as a frequent need for truncating the Newton step, conditioning issues of the associated matrices and convergence failure. The approach presented in this chapter overcomes these shortcomings and finds a major application within the fully automated generation of spline-based parameterizations for *twin-screw machine extruders* (see Chapter 7). It enables improving the methodology from Section 2.10.3 and Chapter 6, which requires finding a *splitting curve* which connects both CUSP points of a twin-screw machine geometry and is strictly contained within the interior of the physical domain. Here, this curve establishes itself as part of the solution to a PDE problem. Furthermore, the proposed framework is compatible with domains comprised of several unit quadrilaterals.

The first step towards applying Isogeometric Analysis (IGA) techniques for the approximate solution of PDE problems is generating an analysis-suitable mapping

operator between parametric and physical domains, using one or several patches, from no more than a (spline-based) description of the boundary contours of the planar physical domain. A subclass of the multitude of the available parameterization algorithms are those based on the principles of *Elliptic Grid Generation* (EGG) which, in their most basic form, attempt to approximate a mapping operator whose inverse is composed of harmonic functions. The main challenge lies in finding a problem formulation that is suitable for a computational approach. A common strategy is approximating the mapping by means of approximately solving a PDE problem. PDE-based EGG has been successfully applied in classical finite difference and finite volume settings and first generalization attempts to spline-based descriptions (as is mandatory in IGA) have been made. Unfortunately, all of the practically viable PDE-based approaches impose certain requirements on the employed spline basis, in particular global $C^{\geq 1}$ continuity.
This chapter discusses an EGG algorithm for the generation of planar parameterizations with locally reduced smoothness (i.e., with support for spline bases with global C^0 continuity). A major use case of the proposed algorithm is that of multipatch parameterizations, made possible by the support for reduced regularity. We propose a specially-tailored solution algorithm that exploits many characteristics of the PDE problem and is suitable for large-scale applications. It is discussed for the singlepatch case before generalizing its concepts to multipatch settings. This chapter is concluded with three numerical experiments and a discussion of the results.

3.1. INTRODUCTION

The automated generation of analysis-suitable planar parameterizations for IGA-based numerical simulation is a challenging, yet important problem as generally no more than a spline- or NURBS-based description of the boundary contours is available. Hence, the main challenge lies in generating a folding-free (i.e., bijective) parameterization of the interior with numerically favorable properties such as orthogonal isolines and a large degree of parametric *smoothness*. Furthermore, a practical algorithm should be computationally inexpensive, and, if possible, exhibit little sensitivity to small perturbations in the parametric description of the boundary contours.
Let Ω denote the open, simply connected target domain and let $\hat{\Omega}$ be an open parametric domain, topologically equivalent to Ω. Furthermore, let $\mathbf{x}_h : \hat{\Omega} \to \Omega$ denote the mapping operator that we attempt to build from the finite-dimensional spline space $\mathcal{V}_h$ with basis $\mathscr{B}_h = \{w_1, w_2, \dots, w_N\}$. In the following, the operator $[\cdot]$ acting on a finite-dimensional spline space returns its canonical basis, which we assume to be clear from context. Hence, we have $[\mathcal{V}_h] = \mathscr{B}_h$.
Here, $\mathbf{x}_h$ extended to the closure of $\hat{\Omega}$ satisfies the relation $\mathbf{x}_h|_{\partial\hat{\Omega}} = \partial\Omega$. Note that $\mathbf{x}_h$ is of the form:

$$\mathbf{x}_h(\xi,\eta) = \sum_{j\in\mathscr{I}_B} \mathbf{z}_j\, w_j(\xi,\eta) + \sum_{i\in\mathscr{I}_I} \mathbf{z}_i\, w_i(\xi,\eta), \tag{3.1}$$

where $\mathscr{I}_I$ and $\mathscr{I}_B$ denote the index-sets of the vanishing and nonvanishing basis functions on $\partial\hat{\Omega}$, respectively. Formally, $\mathscr{I}_B \cap \mathscr{I}_I = \emptyset$ and $\mathscr{I}_B \cup \mathscr{I}_I = \{1,\dots,N\}$. With this,

the objective of all parameterization algorithms is properly selecting the inner control points $\mathbf{z}_i \in \mathbb{R}^2$, while the boundary control points $\mathbf{z}_j \in \mathbb{R}^2$ are known from the boundary correspondence and typically held fixed.
In [GENN12], Gravesen et al. study planar parameterization techniques based on the constrained minimization of a quality functional over the inner control points. To avoid self-intersections, a nonlinear and nonconvex sufficient condition for $\det J > 0$, where J denotes the Jacobian of the mapping, is added as a constraint. The numerical quality of the resulting parameterization depends on the choice of the employed cost functional and the characteristic properties of Ω. While this approach is not guaranteed to yield acceptable results for all types of geometries (see section 3.4), it is known to yield good results in a wide range of applications with proper parameter tuning. A drawback is the relatively large number of required iterations (typically ~ 30) and the need to find an initial guess that also satisfies the constraints (for which another optimization problem has to be solved first). The proposed minimization is tackled with a blackbox nonlinear optimizer (such as IPOPT [BZ09]) that comes with all the drawbacks of nonlinear, nonconvex optimization such as the danger of getting stuck in local minima.
Another class of parameterization methods suitable for nontrivial geometries are PDE-based, most notably, the class of methods based on the principles of *Elliptic Grid Generation* (EGG). Methods based on EGG attempt to generate a mapping $\mathbf{x} : \hat{\Omega} \to \Omega$ such that the components of $\mathbf{x}^{-1} : \Omega \to \hat{\Omega}$ are harmonic functions on Ω. For this, a nonlinear partial differential equation is imposed on $\mathbf{x}$. Let $\boldsymbol{\xi} = (\xi, \eta)^T$ denote the local coordinates in $\hat{\Omega}$. The PDE problem takes the form

$$\mathscr{L}(\mathbf{x}) = g_{22}\mathbf{x}_{\xi\xi} - 2g_{12}\mathbf{x}_{\xi\eta} + g_{11}\mathbf{x}_{\eta\eta} = \mathbf{0}, \quad \text{s.t.} \quad \mathbf{x}|_{\partial\hat{\Omega}} = \partial\Omega, \tag{3.2}$$

with

$$g_{11}(\mathbf{x}) = \mathbf{x}_\xi \cdot \mathbf{x}_\xi, \quad g_{12}(\mathbf{x}) = \mathbf{x}_\xi \cdot \mathbf{x}_\eta \quad \text{and} \quad g_{22}(\mathbf{x}) = \mathbf{x}_\eta \cdot \mathbf{x}_\eta \tag{3.3}$$

the entries of the metric tensor of the mapping (which is nonlinear in $\mathbf{x}$). Given a homeomorphic spline-based boundary correspondence $\partial\hat{\Omega} \to \partial\Omega$ and assuming that $\hat{\Omega}$ is convex, it can be shown that the exact solution of (3.2) is a bijection, justifying a numerical approximation for the purpose of generating a geometry description [Aza09].
EGG has been an established approach in classical meshing for decades and first attempts to apply it to spline-based geometry descriptions were made in [Man89], where the equations are approximately solved by a collocation at the abscissae of a Gaussian quadrature scheme with cubic Hermite splines. In [LB07], the collocation takes place at the Greville abscissae and the resulting nonlinear equations are solved using a Picard-based iterative scheme, allowing for a wider range of spline bases. However, as a downside, the consistency order of Greville-based collocation is not optimal. In [HMV18a], the equations are discretized with a Galerkin scheme and a Newton-based iterative approach is employed for the resulting root-finding problem, allowing for $C^{\geq 1}(\hat{\Omega})$ continuous bases. Numerical convergence is accelerated by generating initial guesses utilizing multigrid techniques and convergence is typically achieved within four (unconstrained) nonlinear iterations.
Unfortunately, none of the aforementioned approaches allow for spline bases with locally reduced (i.e., global $C^0(\hat{\Omega})$) regularity, hence requiring a diffeomorphic boundary

correspondence between the various sides of $\hat{\Omega}$ and Ω. Since in certain applications employing globally $C^0(\hat{\Omega})$ continuous bases is desirable or unavoidable, notably in multipatch settings or when the boundary correspondence is piecewise homeomorphic (but not diffeomorphic), the requirement of higher-order regularity is restrictive. To allow for globally $C^0(\hat{\Omega})$ continuous bases, one may instead minimize the *Winslow functional* [Win81]. Unfortunately, this leads to a formulation in which the Jacobian determinant appears in the denominator. Hence, an iterative solution scheme has to be initialized with a bijective initial guess in order to avoid division by zero, restricting it to use cases in which a bijective initial guess is available (which is usually not the case).

Motivated by our striving for a computationally inexpensive parameterization algorithm that does not require a bijective initial guess and allows for spline spaces with locally reduced regularity, in this chapter, we augment the discretization proposed in [HMV18a] with auxiliary variables, leading to a mixed-FEM-type problem. To allow for large-scale problems, we present a solution strategy that tackles the resulting nonlinear root-finding problem with a Newton-Krylov-based [KK04] Jacobian-free iterative approach that only operates on the nonlinear part (corresponding to the primary, not auxiliary variables) of the equation. Besides singlepatch problems, we will address potential use cases of the algorithm within multipatch settings (in particular with extraordinary vertices), made possible by the support of C^0-continuous spline bases. We conclude this chapter with a number of test cases and a discussion of the results.

3.2. PROBLEM FORMULATION

In [HMV18a], the following discretization of the governing equations (see equation (3.2)) is proposed:

$$\begin{aligned} &\text{find } \mathbf{x}_h \in \mathcal{V}_h^2 \quad \text{s.t.} \\ &\int_{\hat{\Omega}} \boldsymbol{\sigma}_h \cdot \mathscr{L}(\mathbf{x}_h) \mathrm{d}\boldsymbol{\xi} = 0, \quad \forall \boldsymbol{\sigma}_h \in (\mathcal{V}_h^\circ)^2 \quad \text{and} \quad \mathbf{x}_h|_{\partial\hat{\Omega}} = \partial\Omega, \end{aligned} \tag{3.4}$$

where $\mathcal{V}_h^\circ \equiv \mathcal{V}_h \cap H_0^1(\hat{\Omega})$.
Similarly, [HMV18b] introduces a scaled version of (3.4), namely:

$$\begin{aligned} &\text{find } \mathbf{x}_h \in \mathcal{V}_h^2 \quad \text{s.t.} \\ &\int_{\hat{\Omega}} \boldsymbol{\sigma}_h \cdot \tilde{\mathscr{L}}(\mathbf{x}_h) \mathrm{d}\boldsymbol{\xi} = 0, \quad \forall \boldsymbol{\sigma}_h \in (\mathcal{V}_h^\circ)^2 \quad \text{and} \quad \mathbf{x}_h|_{\partial\hat{\Omega}} = \partial\Omega, \end{aligned} \tag{3.5}$$

where

$$\tilde{\mathscr{L}}(\mathbf{x}) = \frac{\mathscr{L}(\mathbf{x})}{\underbrace{g_{11} + g_{22}}_{\geq 0} + \underbrace{\epsilon}_{>0}}. \tag{3.6}$$

Here, $\epsilon > 0$ is a small positive parameter that is usually taken to be $\epsilon = 10^{-4}$. The motivation to solve (3.5) rather than (3.4) is based on the observation that numerical root-finding algorithms typically converge faster and that a suitable convergence criterion is

less geometry-dependent, thanks to scaling invariance. Note that the scaling is allowed because the exact solution is unchanged. Therefore, we base our reformulation of the problem on (3.5).
In order to reduce the highest-order derivatives from two to one, we introduce a new operator in which we replace first and second order derivatives of $\mathbf{x}$ by the zeroth and first order derivatives of $U : \hat{\Omega} \to \mathbb{R}^{2\times 2}$. With $U = [\mathbf{u}, \mathbf{v}]$, we have

$$\mathscr{U}(U) = \frac{\|\mathbf{v}\|^2 \mathbf{u}_\xi - (\mathbf{u}\cdot\mathbf{v})\left(\mathbf{u}_\eta + \mathbf{v}_\xi\right) + \|\mathbf{u}\|^2 \mathbf{v}_\eta}{\|\mathbf{u}\|^2 + \|\mathbf{v}\|^2 + \epsilon}. \tag{3.7}$$

Note that $\mathscr{U}$ satisfies:

$$\mathscr{U}(\partial_{\boldsymbol{\xi}}\mathbf{x}) = \tilde{\mathscr{L}}(\mathbf{x}), \quad \text{where} \quad \left[\partial_{\boldsymbol{\xi}}\mathbf{x}\right]_{i,j} = \frac{\partial \mathbf{x}_i}{\partial \boldsymbol{\xi}_j} \tag{3.8}$$

denotes the differential $\partial\mathbf{x}(\hat{\Omega})$ of $\mathbf{x}$.
A possible reformulation of (3.5) with auxiliary variables reads:

$$\begin{aligned}
&\text{find } (U_h, \mathbf{x}_h) \in \bar{\mathcal{V}}_h^{2\times 2} \times \mathcal{V}_h^2 \text{ s.t.} \\
&\int_{\hat{\Omega}} \Phi_h : (U_h - \partial_{\boldsymbol{\xi}}\mathbf{x}_h)\mathrm{d}\boldsymbol{\xi} + \int_{\hat{\Omega}} \boldsymbol{\sigma}_h \cdot \mathscr{U}(U_h)\mathrm{d}\boldsymbol{\xi} = 0, \quad \forall (\Phi_h, \boldsymbol{\sigma}_h) \in \bar{\mathcal{V}}_h^{2\times 2} \times (\mathcal{V}_h^\circ)^2 \\
&\text{and} \quad \mathbf{x}_h|_{\partial\hat{\Omega}} = \partial\Omega.
\end{aligned} \tag{3.9}$$

Here, $\bar{\mathcal{V}}_h$ spanned by $\left[\bar{\mathcal{V}}_h\right] = \{\bar{w}_1, \dots, \bar{w}_{\bar{N}}\}$, denotes the auxiliary variable basis and $A : B$ the Frobenius inner product between matrices A and B.

Remark. For convenience, we assume that all components of $\partial_{\boldsymbol{\xi}}\mathbf{x}_h$, are projected onto the same auxiliary space $\bar{\mathcal{V}}_h$, hence the appearance of $\bar{\mathcal{V}}_h^{2\times 2}$ in (3.9). However, the framework is also compatible with differing auxiliary bases. In this case, $\bar{\mathcal{V}}_h^{2\times 2}$ is replaced by a suitable tensorial space in (3.9).

Note that the choice of (3.7) is not unique. Here, we have chosen to divide $\mathbf{x}_{\xi\eta}$ equally among $\mathbf{u}_\eta$ and $\mathbf{v}_\xi$. In general, any combination

$$\mathbf{x}_{\xi\eta} \to \chi\mathbf{u}_\eta + (1-\chi)\mathbf{v}_\xi, \tag{3.10}$$

with $\chi \in \mathbb{R}$ is plausible.
System (3.9) now constitutes a discretization of (3.2) that allows for globally $C^{\geq 0}(\hat{\Omega})$-continuous spline spaces at the expense of increasing the problem size from $2 \times |\mathscr{I}_I|$ to $2 \times |\mathscr{I}_I| + 4 \times |\left[\bar{\mathcal{V}}_h\right]|$, where, as before, $\mathscr{I}_I$ refers to the index-set of inner control points.
Let us remark that in certain settings, it suffices to invoke auxiliary variables in one coordinate direction only. Let $U^\xi(\mathbf{u}, \mathbf{v}) = \left[\mathbf{u}, \mathbf{v}_\eta\right]$. A possible problem formulation for the ξ-direction reads:

$$\begin{aligned}
&\text{find } (\mathbf{u}_h, \mathbf{x}_h) \in \bar{\mathcal{V}}_h^2 \times \mathcal{V}_h^2 \text{ s.t.} \\
&\int_{\hat{\Omega}} \boldsymbol{\phi}_h \cdot \left(\mathbf{u}_h - \frac{\partial \mathbf{x}_h}{\partial \xi}\right)\mathrm{d}\boldsymbol{\xi} + \int_{\hat{\Omega}} \boldsymbol{\sigma}_h \cdot \mathscr{U}\left(U^\xi(\mathbf{u}_h, \mathbf{x}_h)\right)\mathrm{d}\boldsymbol{\xi} = 0, \quad \forall (\boldsymbol{\phi}_h, \boldsymbol{\sigma}_h) \in \bar{\mathcal{V}}_h^2 \times \left(\mathcal{V}_h^\circ\right)^2 \\
&\text{and} \quad \mathbf{x}|_{\partial\hat{\Omega}} = \partial\Omega,
\end{aligned} \tag{3.11}$$

and similarly for the η-direction.
The above approach is useful when C^0-continuities only need to be introduced in a single coordinate direction. In such cases, an approach based on (3.11) reduces the total number of degrees of freedom (DOFs).

3.3. SOLUTION STRATEGY

Systems (3.9) and (3.11) are nonlinear and have to be solved with an iterative algorithm. We will discuss a solution algorithm that is loosely based on the Newton-based approach proposed in [HMV18a] (see Chapter 2). However, we tweak it in order to reduce computational costs and memory requirements by exploiting many characteristics of the problem at hand. First, we discuss the case in which $\hat{\Omega}$ is given by a single patch, after which we generalize our solution strategy to multipatch settings (in particular, allowing for topologies that contain extraordinary vertices).

3.3.1. SINGLEPATCH PARAMETERIZATIONS

With (3.1) in mind, we may write

$$\mathbf{x}_h(\mathbf{c}) = \mathbf{x}_h^\circ(\mathbf{c}) + \mathbf{x}_D, \quad \text{where} \quad \mathbf{x}_h^\circ(\mathbf{c}) \in \left(\mathcal{V}_h^\circ\right)^2 \tag{3.12}$$

with $\mathbf{c} \in \mathbb{R}^{2N_0}$ a vector of unknowns with respect to $[\left(\mathcal{V}_h^\circ\right)^2]$, referring to the $\mathbf{z}_i \in \mathbb{R}^2$ from (3.1). Here, N_0 denotes the cardinality of $\left[\mathcal{V}_h^\circ\right]$, while

$$\mathbf{x}_D = \sum_{j \in \mathscr{I}_B} \mathbf{z}_j w_j \tag{3.13}$$

is the Dirichlet extension of the boundary condition, c.f. (3.1). Meanwhile

$$U_h \longrightarrow U_h(\mathbf{d}) \in \bar{\mathcal{V}}_h^{2\times 2}, \tag{3.14}$$

where $\mathbf{d} \in \mathbb{R}^{4\tilde{N}}$ is an unknown vector of weights with respect to the basis spanning $\bar{\mathcal{V}}_h^{2\times 2}$. With (3.12) and (3.14), we can reinterpret (3.9) as a problem in $\mathbf{c}$ and $\mathbf{d}$. This leads to a residual vector of the form

$$\mathbf{R}(\mathbf{d},\mathbf{c}) = \begin{pmatrix} \mathbf{R}_L(\mathbf{d},\mathbf{c}) \\ \mathbf{R}_{NL}(\mathbf{d}) \end{pmatrix}, \tag{3.15}$$

where $\mathbf{R}_L \in \mathbb{R}^{4\tilde{N}}$ refers to the linear part in (3.9) (the projection of $\mathbf{x}_\xi$ and $\mathbf{x}_\eta$ onto the auxiliary spline space) and $\mathbf{R}_{NL} \in \mathbb{R}^{2N_0}$ to the nonlinear (the term involving $\mathscr{U}$).
The Newton approach from [HMV18a] requires the assembly of the Jacobian

$$J_R = \begin{pmatrix} \frac{\partial \mathbf{R}_L}{\partial \mathbf{d}} & \frac{\partial \mathbf{R}_L}{\partial \mathbf{c}} \\ \frac{\partial \mathbf{R}_{NL}}{\partial \mathbf{d}} & \emptyset \end{pmatrix} \equiv \begin{pmatrix} A & B \\ C & \emptyset \end{pmatrix} \tag{3.16}$$

of (3.9) during every iteration. The matrices A and B, corresponding to the linear part in (3.9), are not a function of $\mathbf{c}$ and $\mathbf{d}$ and thus have to be assembled only once. In fact,

A is block diagonal with blocks given by the parametric mass matrix $\bar{M}$ over the basis $[\bar{\mathcal{V}}_h] = \{\bar{w}_1, \ldots, \bar{w}_{\bar{N}}\}$. We have:

$$\bar{M} = \left(\bar{\boldsymbol{\phi}}, \bar{\boldsymbol{\phi}}^T\right)_{L^2(\hat{\Omega})} \quad \text{and} \quad A = I^{4\times 4} \otimes \bar{M}, \tag{3.17}$$

where $I^{n\times n}$ denotes the identity matrix in $\mathbb{R}^n$ and $\bar{\boldsymbol{\phi}}_i = \bar{w}_i \in [\bar{\mathcal{V}}_h]$.
Likewise, B is block diagonal with blocks

$$\bar{M}^{\xi} = \left(\bar{\boldsymbol{\phi}}, \boldsymbol{\phi}_{\xi}^{\circ T}\right)_{L_2(\hat{\Omega})} \quad \text{and} \quad \bar{M}^{\eta} = \left(\bar{\boldsymbol{\phi}}, \boldsymbol{\phi}_{\eta}^{\circ T}\right)_{L_2(\hat{\Omega})}, \tag{3.18}$$

where $\boldsymbol{\phi}^{\circ}$ is a vector containing the $w_i \in [\mathcal{V}_h^{\circ}]$. We have:

$$B = \begin{pmatrix} -\bar{M}^{\xi} & & & \\ & -\bar{M}^{\xi} & & \\ & & -\bar{M}^{\eta} & \\ & & & -\bar{M}^{\eta} \end{pmatrix}. \tag{3.19}$$

For given $\mathbf{c}$ and $\mathbf{d}$, the Newton search direction is computed as the solution of a system of the form

$$\begin{pmatrix} A & B \\ C & \emptyset \end{pmatrix} \begin{pmatrix} \delta\mathbf{d} \\ \delta\mathbf{c} \end{pmatrix} = \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix}, \tag{3.20}$$

where $C = C(\mathbf{d})$ is, unlike A and B, not constant and has to be reassembled in each iteration. We form the Schur complement of A, in order to yield an equation for $\delta\mathbf{c}$ only, namely:

$$\underbrace{CA^{-1}B}_{D}\delta\mathbf{c} = CA^{-1}\mathbf{a} - \mathbf{b}. \tag{3.21}$$

In order to avoid the computationally expensive assembly of D, we solve (3.21) with a Newton-Krylov [KK04] algorithm which only requires evaluating matrix-vector products of the form $D\mathbf{v}$. They can be approximated with finite differences rather than explicit assembly of D. Since

$$C\mathbf{s} = \frac{\mathbf{R}_{NL}(\mathbf{d} + \epsilon\mathbf{s}) - \mathbf{R}_{NL}(\mathbf{d})}{\epsilon} + \mathcal{O}(\epsilon), \tag{3.22}$$

we have

$$D\mathbf{s} \simeq \frac{\mathbf{R}_{NL}(\mathbf{d} + \epsilon A^{-1}B\mathbf{s}) - \mathbf{R}_{NL}(\mathbf{d})}{\epsilon} \tag{3.23}$$

and

$$CA^{-1}\mathbf{a} \simeq \frac{\mathbf{R}_{NL}(\mathbf{d} + \epsilon A^{-1}\mathbf{a}) - \mathbf{R}_{NL}(\mathbf{d})}{\epsilon}, \tag{3.24}$$

for ϵ small. The optimal choice of ϵ is discussed in [KK04].
We compute products of the form $\mathbf{q} = A^{-1}\mathbf{t}$ from the solution of the system $A\mathbf{q} = \mathbf{t}$, which

has for $\mathbf{t} = B\mathbf{s}$ (see equation (3.23)) and $\mathbf{t} = \mathbf{a}$ (see equation (3.24)) the form of a (separable) L_2-projection.
Product $\mathbf{q} = A^{-1}B\mathbf{s}$ satisfies

$$\mathbf{q}(\mathbf{s}) = \underset{\mathbf{h}}{\operatorname{argmin}} \frac{1}{2} \int_{\hat{\Omega}} \left\| U_h(\mathbf{h}) - \partial_{\boldsymbol{\xi}} \mathbf{x}_h^\circ(\mathbf{s}) \right\|^2 \mathrm{d}\boldsymbol{\xi}, \tag{3.25}$$

and similarly for $\mathbf{q} = A^{-1}\mathbf{a}$.
As such, A is block diagonal and composed of separable mass matrices $\bar{M} = \bar{m}_\xi \otimes \bar{m}_\eta$

$$A = \begin{pmatrix} \bar{m}_\xi \otimes \bar{m}_\eta & & \\ & \ddots & \\ & & \bar{m}_\xi \otimes \bar{m}_\eta \end{pmatrix}, \tag{3.26}$$

where $\bar{m}_\xi$ and $\bar{m}_\eta$ refer to the univariate mass matrices resulting from the tensor-product structure of $[\bar{\mathcal{V}}_h]$. Therefore, we have

$$A^{-1} = \begin{pmatrix} (\bar{m}_\xi^{-1}) \otimes (\bar{m}_\eta^{-1}) & & \\ & \ddots & \\ & & (\bar{m}_\xi^{-1}) \otimes (\bar{m}_\eta^{-1}) \end{pmatrix}. \tag{3.27}$$

We follow the methodology from [GC14], where a computationally inexpensive inversion of the bivariate mass matrix (which is never explicitly assembled) is achieved by repeated inversion with the univariate mass matrices $\bar{m}_\xi$ and $\bar{m}_\eta$. Here, we use a direct approach by computing Cholesky decompositions [SS89]. An inversion can be done in only $\mathcal{O}(\bar{N})$ arithmetic operations and Cholesky decompositions have to be formed only once, thanks to the fact that A is constant. Similarly, we can exploit the separable nature of the blocks that make up B in order to efficiently multiply with B.
After solving (3.21), $\delta\mathbf{d}$ is found as the solution of

$$A\delta\mathbf{d} = \mathbf{a} - B\delta\mathbf{c}. \tag{3.28}$$

Upon completion, the vector $\mathbf{n} \equiv (\delta\mathbf{d}, \delta\mathbf{c})^T$ constitutes the Newton search direction. We update the current iterate $(\mathbf{d}, \mathbf{c})^T$ by adding $\kappa\mathbf{n}$, where the optimal value of $\kappa \in (0, 1]$ is estimated through a line search routine. Above steps are repeated until the norm of $\mathbf{n}$ is negligibly small. Upon completion, we extract the $\mathbf{c}$-component from the resulting solution vector which contains the inner control points of the mapping operator $\mathbf{x}_h$, while the $\mathbf{d}$-component serves no further purpose and can be discarded.
It should be noted that a single matrix-vector product $D\mathbf{s}$ is slightly more expensive than, for instance, $C\mathbf{s}$, due to the requirement to invert A. However, thanks to the separable nature of A (and B), the costs in (3.23) are dominated by function evaluations in $\mathbf{R}_{NL}$. In general, we have to regard $\mathcal{U}(U_h)$ as dense (i.e., not compactly supported in $\hat{\Omega}$). However, $\mathcal{U}(U_h)$ is tested against compactly supported functions $\boldsymbol{\sigma}_h \in (\mathcal{V}_h^\circ)^2$, c.f. (3.9). Hence, the assembly costs of $\mathbf{R}_{NL}(\mathbf{d})$ are of the same order as assembling the nonlinear residual corresponding to an approach without auxiliary variables. Note, however, that

if the space $\bar{\mathcal{V}}_h$ is h-refined with respect to $\mathcal{V}_h$, $\mathbf{R}_{NL}$ has to be assembled over the finer element subdivision of $\hat{\Omega}$, which means additional computational costs. In practice, an implementation that exploits all discussed characteristics of the problem formulation is somewhat slower than applying Newton-Krylov to an $H^2(\hat{\Omega})$-conforming discretization. However, we regard this as a minor shortcoming given the additional potential use cases, made possible by the reduced continuity requirements.
There exist many possible choices of constructing an initial guess for the **c**-component of the iterative scheme. Common choices are algebraic methods, most notably transfinite interpolation [GH73]. Once the **c**-component has been computed with one of the available methods, a reasonable way to compute the corresponding **d**-part is through a (separable) projection of $\partial_{\boldsymbol{\xi}}\mathbf{x}_h$ onto $\bar{\mathcal{V}}_h^{2\times 2}$.
A slightly superior initial guess can be generated using multigrid techniques as demonstrated in [HMV18a]. The problem is first solved using a coarser basis and an algebraic initial guess, after which the coarse solution vector is prolonged and subsequently used as an initial guess. This is compatible with the techniques discussed in this section. However, instead of prolonging the full coarse-grid solution vector, we only prolong the **c**-component and compute the corresponding **d**-component using an $L_2(\hat{\Omega})$-projection to ensure that the residual vector evaluated in the initial guess equals zero on the entries that correspond to the projection onto the auxiliary spline space.

3.3.2. MULTIPATCH

The reformulation with auxiliary variables has a particularly interesting application within multipatch settings, especially when extraordinary patch vertices are present. Most of the techniques from subsection 3.3.1 are readily applicable but there exist subtle differences which we address in the following.
Let $\hat{\Omega}$ be an open, convex, polygonal multipatch domain. We assume that $\hat{\Omega}$ is covered by the closure of a set of mutually disjoint images $\{\hat{\Omega}_1,\dots\hat{\Omega}_n\}$ of the unit quadrilateral $\tilde{\Omega} = (0,1)^2$ under the bijective mappings $\hat{\mathbf{m}}_i : \tilde{\Omega} \to \hat{\Omega}_i,\ \ i \in \{1,\dots,n\}$. Hence,

$$\hat{\Omega} = \mathrm{Int}\left(\bigcup_{i=1}^{n} \overline{\hat{\Omega}}_i\right), \tag{3.29}$$

where $\overline{\hat{\Omega}}_i$ denotes the closure of $\hat{\Omega}_i$ and $\mathrm{Int}(\cdot)$ the interior of a closed domain.
For convenience, we assume that each $\hat{\Omega}_i$ is an affine transformation of the reference unit quadrilateral $\tilde{\Omega}$. Therefore

$$\hat{\mathbf{m}}_i(\boldsymbol{\mu}) = Q_i\boldsymbol{\mu} + \mathbf{b}_i, \tag{3.30}$$

where $Q_i \in \mathbb{R}^{2\times 2}$ is an invertible (and orientation preserving) matrix, $\mathbf{b}_i \in \mathbb{R}^2$ some translation and the vector $\boldsymbol{\mu} = (\mu,\nu)^T$ contains the free local coordinate functions in $\tilde{\Omega}$.

Remark. The automated generation of a multipatch structure is a nontrivial task, which is not discussed in this chapter. For an overview of possible segmentation techniques, which can be applied when a multipatch covering of $\hat{\Omega}$ is not self-evident or given, we refer to [BJ17, XKFC18, FJ19].

Let $\mathbf{x}:\hat{\Omega}\to\Omega$ be such that $\mathbf{x}^{-1}:\Omega\to\hat{\Omega}$ is composed of harmonic functions on Ω. Given a convex $\hat{\Omega}$ and assuming that the boundary correspondence $\mathbf{x}_D:\partial\hat{\Omega}\to\partial\Omega$ is homeomorphic and Ω has no concave corners, Rado's theorem [SY97] applies and a harmonic $\mathbf{x}^{-1}$ is bijective.
In the case of multipatch, pairs of faces $(\gamma_i^\alpha,\gamma_j^\beta)\subset\partial\hat{\Omega}_i\times\partial\hat{\Omega}_j$ and sets of vertices $\{\mathbf{p}_i,\dots,\mathbf{p}_l\}\subset\partial\hat{\Omega}_i\times\ldots\times\partial\hat{\Omega}_l$ may coincide on $\hat{\Omega}$. The objective is to build primal and auxiliary spline spaces with elements from $H^1(\hat{\Omega})$. We construct such spaces starting from the patchwise discontinuous spaces $\mathcal{V}_{h,i}^{\text{disc}}$ and $\bar{\mathcal{V}}_{h,i}^{\text{disc}}$, acquired upon performing a push-forward of the locally defined spline bases $\mathscr{B}_{h,i}^{\text{loc}}$ and $\bar{\mathscr{B}}_{h,i}^{\text{loc}}$ with elements from $H^1(\tilde{\Omega})$. We have:

$$\left[\mathcal{V}_{h,\text{disc}}\right]=\bigcup_{i=1}^{n}\left\{W_j\circ\hat{\mathbf{m}}_i^{-1}\;\middle|\;W_j\in\mathscr{B}_{h,i}^{\text{loc}}\right\}\tag{3.31}$$

and similarly for $\bar{\mathcal{V}}_{h,\text{disc}}$. The spaces $\mathcal{V}_h$ and $\bar{\mathcal{V}}_h$ then follow from taking the intersections of $\mathcal{V}_{h,i}^{\text{disc}}$ and $\bar{\mathcal{V}}_{h,i}^{\text{disc}}$ with $C^0(\hat{\Omega})$.

Remark. The $w_i\in[\mathcal{V}_h]$ (and similarly for $\bar{\mathcal{V}}_h$) are conveniently generated by identifying the indices of functions from $\left[\mathcal{V}_{h,\text{disc}}\right]$ that coincide on the interfaces of $\hat{\Omega}$. They are then coupled in order to ensure that all $w_i\in\mathcal{V}_h$ are single-valued on $\hat{\Omega}$ by default. Clearly, for this to be possible, the knot vectors used for neighbouring patches need to be compatible, which we assume to be the case.

In the multipatch setting, we solve (3.9) by evaluating the associated integrals through a sequence of pullbacks from the $\hat{\Omega}_i\subset\hat{\Omega}$ into the reference domain $\tilde{\Omega}$. Thanks to the affine nature of the pullback, replacing $\boldsymbol{\xi}$-derivatives by local $\boldsymbol{\mu}$-derivatives is straightforward. As such, the solution of (3.9) is associated with a collection of mappings $\{\tilde{\mathbf{x}}_h^1,\dots,\tilde{\mathbf{x}}_h^n\}$, with $\tilde{\mathbf{x}}_h^i:\tilde{\Omega}\to\Omega_i\subset\Omega$, whose elements satisfy

$$\tilde{\mathbf{x}}_h^i\simeq\mathbf{x}|_{\hat{\Omega}_i}\circ\hat{\mathbf{m}}_i.\tag{3.32}$$

As the right hand side of (3.32) is a composition of bijective mappings, the bijectivity of the $\tilde{\mathbf{x}}_h^i$ is conditional on the quality of the approximation $\mathbf{x}_h:\hat{\Omega}\to\Omega$ of $\mathbf{x}$. If the $\tilde{\mathbf{x}}_h^i$ are bijective, they jointly form a parameterization of Ω.
Unlike in the singlepatch setting, the $L_2(\hat{\Omega})$-projection associated with the linear part of the residual vector is not separable (unless the $\hat{\Omega}_i$ form a structured topology). As such, the evaluation of vector products $A^{-1}B\mathbf{s}$ (see equation (3.23)) becomes more involved. A possible workaround is explicit assembly and inversion of the Jacobian of the system (see equation (3.20)), leading to increased computational times and memory requirements.
A possible alternative is approximating products of the form $A^{-1}B\mathbf{s}$ by a sequence of patchwise separable operations. In the following, we sketch a plausible approach.
Similar to the singlepatch case, products of the form $\mathbf{q}=A^{-1}B\mathbf{s}$ satisfy

$$\mathbf{q}(\mathbf{s})=\underset{\mathbf{h}}{\text{argmin}}\frac{1}{2}\sum_{i=1}^{n}\int_{\hat{\Omega}_i}\left\|U_h(\mathbf{h})-\partial_{\boldsymbol{\xi}}\mathbf{x}_h^\circ(\mathbf{s})\right\|^2\mathrm{d}\boldsymbol{\xi}.\tag{3.33}$$

Let

$$U_h^{\text{disc}}(\mathbf{q}_{\text{disc}}) \in \bar{\mathcal{V}}_{h,\text{disc}}^{2\times 2}, \tag{3.34}$$

where $\mathbf{q}_{\text{disc}}$ is defined with respect to $\left[\bar{\mathcal{V}}_{h,\text{disc}}^{2\times 2}\right]$.
In order to approximate $\mathbf{q} = A^{-1}B\mathbf{s}$, we select $\mathbf{q}_{\text{disc}}$ as the solution of:

$$\mathbf{q}_{\text{disc}}(\mathbf{s}) = \underset{\mathbf{h}}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{n} \int_{\hat{\Omega}_i} \left\| U_h^{\text{disc}}(\mathbf{h}) - \partial_{\boldsymbol{\xi}} \mathbf{x}_h^{\circ}(\mathbf{s}) \right\|^2 \mathrm{d}\boldsymbol{\xi}. \tag{3.35}$$

We perform a patchwise pullback of the L_2-projections into the reference domain where they are solved with the techniques from Section 3.3.1. Thanks to the affine nature of the pullback, the geometric factor associated with $\hat{\Omega}_i$ is constant and given by

$$\det J_i = \det Q_i. \tag{3.36}$$

Therefore, separability is not lost and the same efficiency as in the singlepatch case is achieved.

Remark. Requiring that each $\hat{\mathbf{m}}_i : \tilde{\Omega} \to \hat{\Omega}_i$ is affine may be too restrictive in practice. In general, any transformation whose geometrical factor is of the form $\det J_i(\mu, \nu) = f_i(\mu) g_i(\nu)$ may be used. The individual factors then serve as the geometrical factors in the assembly of the univariate mass matrices that correspond to the projection from (3.35). By this, separability is retained.

We restrict the solution of (3.35) to $\bar{\mathcal{V}}_h^{2\times 2}$ by performing a weighted sum of components that coincide under coupling. Let $\bar{\Phi}_i \in \left[\bar{\mathcal{V}}_h^{2\times 2}\right]$ result from the coupling of $\{\tilde{\Phi}_\alpha, \ldots, \tilde{\Phi}_\gamma\} \subset \bar{\mathcal{V}}_{\text{h, disc}}^{2\times 2}$ and let the elements of the set $\{S_\alpha, \ldots, S_\gamma\}$ satisfy

$$S_\beta = \sum_{ij} \int_{\hat{\Omega}} \left[\bar{\Phi}_\beta\right]_{ij} \mathrm{d}\boldsymbol{\xi}, \quad \bar{\Phi}_\beta \in \{\tilde{\Phi}_\alpha, \ldots, \tilde{\Phi}_\gamma\}. \tag{3.37}$$

If the $\{\tilde{\Phi}_\alpha, \ldots, \tilde{\Phi}_\gamma\}$ receive the weights $g_\alpha, \ldots, g_\gamma$ under the projection from (3.35), we set

$$\mathbf{q}_i = \frac{S_\alpha g_\alpha + \ldots + S_\gamma g_\gamma}{S_\alpha + \ldots + S_\gamma}. \tag{3.38}$$

Relation (3.38) induces a canonical restriction operator from $\bar{\mathcal{V}}_{\text{h, disc}}^{2\times 2}$ to $\bar{\mathcal{V}}_{\text{h}}^{2\times 2}$ that is used to approximate $\mathbf{q}$ from $\mathbf{q}_{\text{disc}}$.

3.4. NUMERICAL EXPERIMENTS

In the following, we present several numerical experiments, demonstrating the functioning of the proposed algorithm. First, we present two singlepatch problems after which we present a more involved multipatch parameterization.
In all cases, the auxiliary variable spline space $\bar{\mathcal{V}}_h$ results from one uniform h-refinement of $\mathcal{V}_h$.

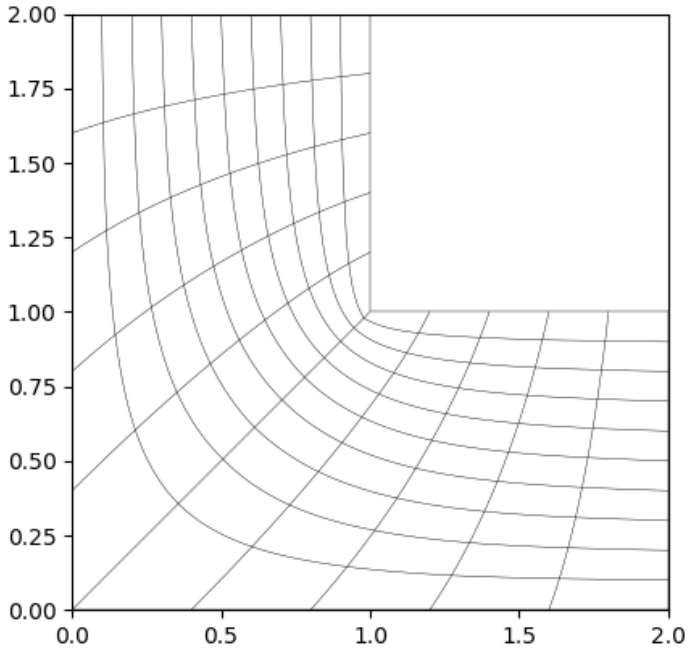


Figure 3.1: Solution of the L-bend problem with the mixed-FEM algorithm.

3.4.1. L-BEND

As a proof of concept, we present results for the well-known singlepatch L-bend problem. Wherever possible, we shall compare the results to a direct minimization of the Winslow functional

$$W(\mathbf{x}) = \int_{\hat{\Omega}} \frac{g_{11} + g_{22}}{\det J} \mathrm{d}\boldsymbol{\xi}, \tag{3.39}$$

whose global minimizer (over $\mathcal{V}_h^2$, with appropriate boundary conditions) coincides with a numerical approximation of the solution of (3.2) in the limit where $N \to \infty$ [Aza09]. For the L-bend problem, we employ uniform knot vectors in both directions with a p-fold knot repetition at $\xi = 0.5$ in order to properly resolve the C^0-continuity. Here, we utilize a bicubic basis, i.e., $p = 3$. Since a repeated knot is only introduced in ξ-direction, we solve (3.11) rather than (3.9). Figure 3.1 shows the resulting parameterization along with the element boundaries under the mapping. The Schur complement solver converges after 3 iterations which amounts to 106 evaluations of $\mathbf{R}_{NL}$. As can be seen in the figure, the parameterization is symmetric across the straight line connecting the upper and lower C^0-continuities, which is expected behaviour from the shape of the geometry. We

regard this as a positive sanity check for the functioning of the algorithm. Furthermore, we observe that despite the presence of knot repetitions at $\xi = 0.5$, the parameterization is highly regular in the interior. Again, this is a positive result since the solution is expected to be an approximation of the global minimizer of (3.39) (over $\mathcal{V}_h^2$), which, in turn, approximates a mapping that is diffeomorphic in the interior. A substitution of the solution vector $\mathbf{c}_{\text{PDE}}$ of the system of equations (3.11) in (3.39) gives

$$W(\mathbf{c}_{\text{PDE}}) \simeq 3.01518, \tag{3.40}$$

whereas the global minimizer $\mathbf{c}_{\text{W}}$ of (3.39) over the same basis yields

$$W(\mathbf{c}_{\text{W}}) \simeq 3.01425. \tag{3.41}$$

This constitutes another positive sanity check as the results are very close, while a substitution of the PDE solution is slightly above the global minimum. Hence, the PDE problem trades the possibility of using a degenerate initial guess for an increased problem size and a slight reduction in parameterization quality, as measured by (3.39). Note that (3.39) is compatible with bases of reduced regularity by default (without the need to introduce auxiliary variables).
Hence, the PDE solution comes with all the undesired characteristics of inversely harmonic maps such as the tendency to yield bundled / spread isolines near concave / convex corners. This does not occur in parameterizations based on the techniques of [GENN12] (see Figure 3.2). However, the L-bend example is rather contrived since a good parameterization is easily constructed with algebraic techniques. Here, the results only serve as a proof of concept.

Remark. We address the mitigation of aforementioned undesirable features in Chapter 4.

3.4.2. TUBE-LIKE SHAPED GEOMETRY

In many cases, segmentation along knots with p-fold repetition and continuation with, for instance, techniques from [HMV18a] on the smaller pieces is a viable choice. However, in some cases, a segmentation curve along which to split the geometry into smaller parts may be hard to find. One such example is depicted in Figure 3.3 (left), which is a geometry taken from the practical application of numerically simulating a twin-screw machine. For convenience, the $\xi = 0.5$ isoline, across which the mapping is C^0-continuous, has been plotted in red. The usefulness of the proposed algorithm becomes apparent in this case: instead of having to generate a valid $\xi = 0.5$ isoline, the isoline establishes itself from the solution of the PDE problem.
As in the L-bend problem, we observe that the resulting parameterization is highly regular across the $\xi = 0.5$ isoline, despite the continuity properties of $\mathcal{V}_h$ and the spiked upper and lower boundaries.
The proposed algorithm produces superior results to the constrained optimization approach from [GENN12] (see Figure 3.3, right). In fact, here we initialized the optimization by the PDE solution, as the solver struggles to find a feasible initial guess through optimization. This confirms the finding from [HMV18a] that EGG-based approaches may be

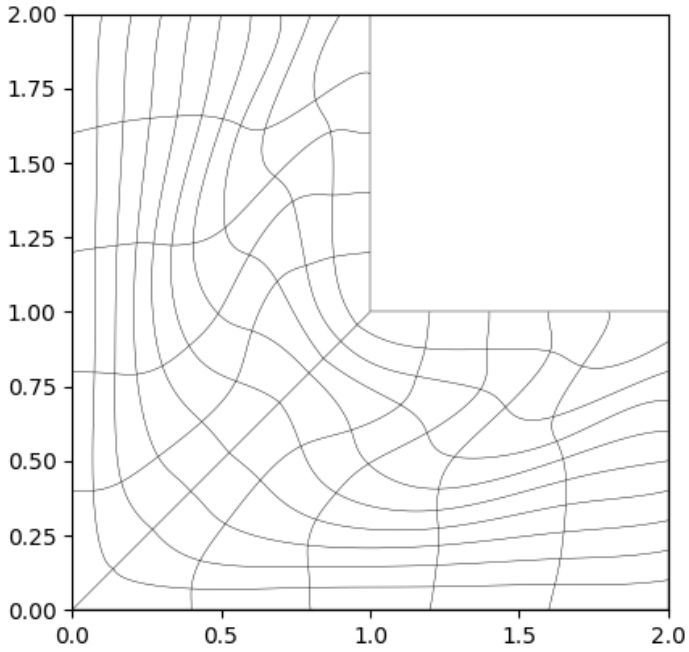


Figure 3.2: Solution of the L-bend problem with constrained minimization of the *Area Orthogonality* functional (see [GENN12]).

a viable alternative to finding feasible initial guesses for approaches based on optimization. Furthermore, we note the striking difference in the required number of iterations, which amount to over 100 (constrained) in the optimization, while the PDE solver converges in only 7 iterations.

The poor performance of the optimization approach can be explained by the tiny gaps of the geometry, leading to natural jumps in the magnitude of the Jacobian determinant. As most cost functions are functions of the g_{ij}, they are very sensitive to jumps in $\det J$. This is further evidenced by the poor grid quality in the narrow part of the geometry (see Figure 3.4, right). In our experience, this is not the case for the PDE solution (see Figure 3.4, left) and we successfully employed the approach for the automatic generation of a large number of similar geometries.

Finally, it should be noted that a comparison to the global minimizer of the Winslow energy is not possible since the gradient-based optimizer we employed failed to further reduce the cost function from the evaluation of the PDE solution.

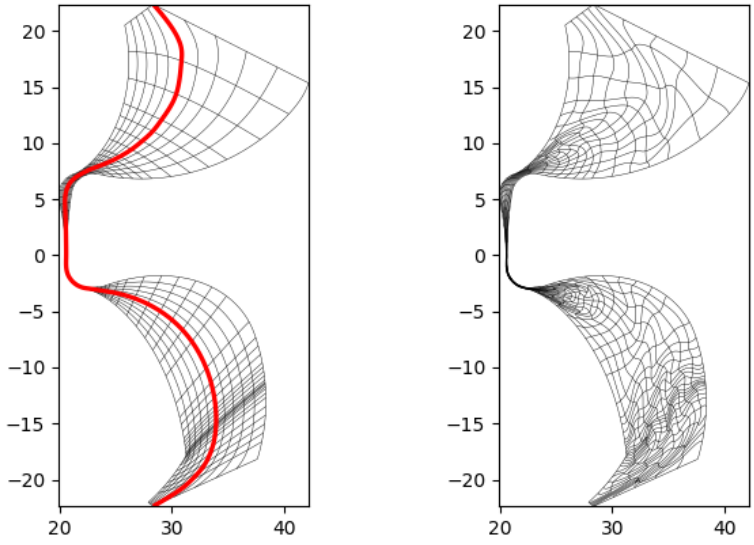


Figure 3.3: PDE-based parameterization (left) and area-orthogonality minimized parameterization (right) of a tube-like shaped geometry.

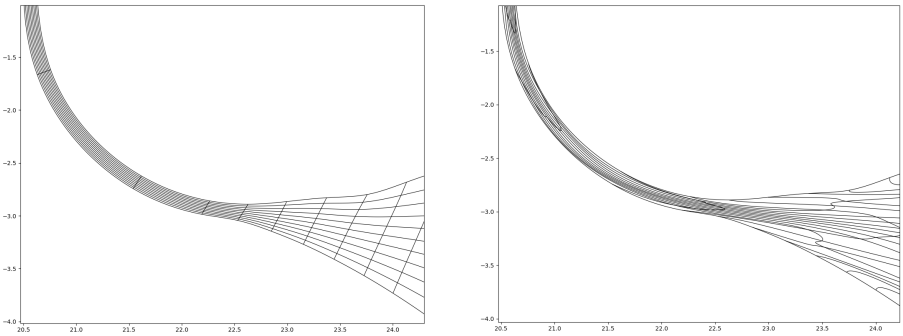


Figure 3.4: Zoom-in on the PDE-based parameterization and area-orthogonality minimization parameterization.

3.4.3. MULTIPATCH PROBLEM - THE BAT GEOMETRY

A further interesting application of the proposed algorithm is that of a multipatch parameterization. In Section 3.4.2, we have successfully employed the algorithm to a geometry with a C^0-continuity across the $\xi = 0.5$ isoline, which might as well be regarded as a two-patch parameterization with coupling along aforementioned interface. Clearly, a topology with an uneven number of patches that contains extraordinary vertices is of major interest. We are considering the diamond-shaped triple-patch domain depicted

in Figure 3.5 (left). The target boundaries form the bat-shaped contour depicted in Figure 3.5 (right). Note that, as required in Section 3.3.2, the parametric domain forms a convex subset of $\mathbb{R}^2$. For convenience we have highlighted the positions of the various boundaries under the mapping in different colors. Of course, of major interest shall be how the dotted red interfaces in Figure 3.5 (left) are deformed under the mapping. Figure

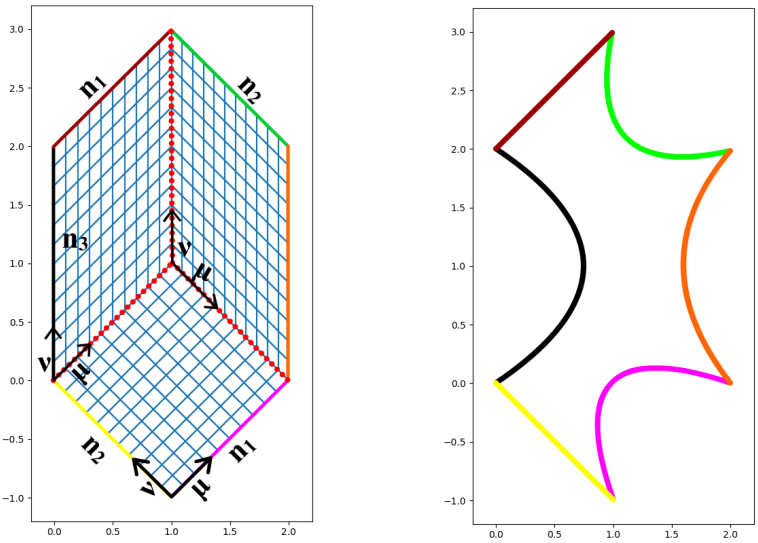


Figure 3.5: Diamond shaped multipatch domain (left) and the target boundaries (right). Here, $n_1 = 10$, $n_2 = 11$ and $n_3 = 12$ denote the number of (uniformly-spaced) elements in each coordinate direction. There are no internal knot repetitions.

3.6 (left) shows the mapping we utilize to initialize the Newton-Krylov solver while Figure 3.6 (right) shows the resulting geometry. Even though better initial guesses are easily constructed, here we have chosen to initialize the solver with a degenerate mapping in order to demonstrate that bijectivity is not a necessary condition for convergence.

Remark. As transfinite interpolation does not readily generalize to multipatch, we typically construct initial guesses by solving a *forward* Laplace problem first. I.e., the components of $\mathbf{x}_h$ approximate harmonic functions in $\hat{\Omega}$.

The Newton-Krylov solver converges after six nonlinear iterations. The dotted red curves in Figure 3.6 (right) show the internal interfaces of $\hat{\Omega}$ under the mapping. We see that the patch interfaces are mapped into the interior of $\overline{\Omega}$. The resulting geometry parameterization is bijective. However, the isolines make steep angles by the internal patch interfaces. This results from the additional pull back of $\tilde{\mathbf{x}}|_{\hat{\Omega}_i}$ into $\tilde{\Omega}$ via the operator $\hat{\mathbf{m}}_i$ (see equation (3.32)), which generally introduces a C^0-continuity in the composite mapping. Higher-order smoothness across patch interfaces is generally difficult to achieve and usually accomplished by constructing bases whose elements possess higher-order continuity sufficiently distant from the extraordinary vertices. However, note that such

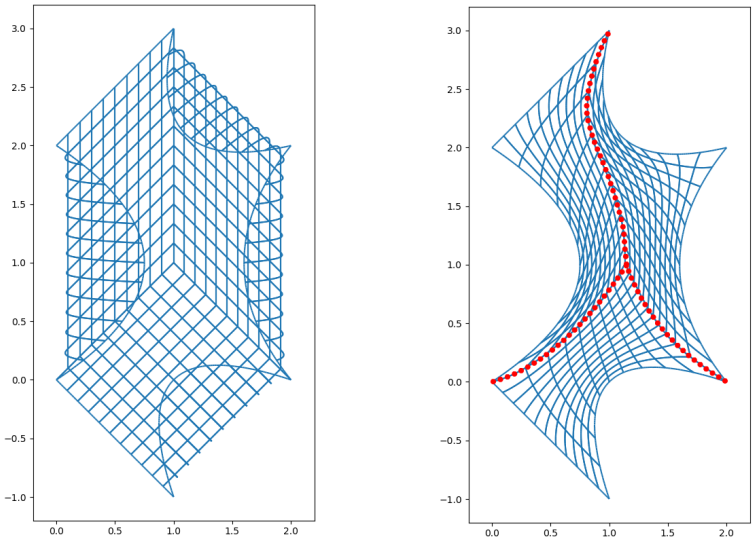


Figure 3.6: The mapping that is passed on to the solver (left) and the resulting parameterization (right).

bases may not allow for patchwise-affine transformations such that $L_2(\hat{\Omega}_i)$-projections may lose their separability property. For strategies to build bases with higher-order regularity across patch interfaces, we refer to [BJM16].

3.5. CHAPTER CONCLUSIONS

We have presented an IGA-suitable EGG algorithm that is compatible with spline bases $\mathcal{V}_h$ possessing reduced regularity (whereby *reduced* stands for global $C^{\geq 0}(\hat{\Omega})$-continuity) by introducing a set of auxiliary variables. We proposed an iterative Newton-Krylov approach operating on the Schur complement of the linear part of the resulting nonlinear system of equations, which operates efficiently and reduces memory requirements. As such, it is suitable for large-scale problems. Unlike similar C^0-compatible EGG-based approaches, the iterative solution method does not have to be initialized with a bijective mapping, significantly improving its usability in practice. However, this major advantage comes at the expense of increasing the problem size. The impact is partially mitigated by the specially-tailored iterative solution algorithm.
We have presented three numerical experiments, two with a single patch and one resulting from a triple-patch configuration. In the singlepatch case, we concluded that a substitution of the PDE solution into the Winslow functional (equation (3.39)) yields an outcome that is close to the global minimizer (over $\mathcal{V}_h^2$), which is generally hard to find through direct minimization, due to the presence of $\det J$ in the denominator, c.f. (3.39). As such, we concluded that the algorithm operates as expected and offers a viable alternative to direct minimization of (3.39). However, it also comes with all the well-documented pathologies of inversely harmonic maps, such as the tendency to yield

bundled / spread isolines near concave / convex corners [Aza09].
Convergence is typically reached within only a few iterations. The required number of iterations can be further reduced by employing multigrid techniques (see [HMV18a]).
A major use case of the proposed algorithm is that of multipatch applications. In Section 3.4.3, we presented results of the application to a triple-patch topology, where we successfully generated a patchwise bijective parameterization by approximating the composition of an inversely harmonic mapping and patchwise affine transformations. The position of internal patch interfaces under the mapping do not have to be imposed manually but follow naturally from the composite PDE solution.
Finally, we observed that the composition with affine transformations results in nonsmooth transitions at patch interfaces. Higher-order regularity across interfaces can be achieved by a clever coupling of inter-patch DOFs sufficiently distant from extraordinary vertices.

REFERENCES

[Aza09] Boris Nikolaevich Azarenok. Generation of structured difference grids in two-dimensional nonconvex domains using mappings. *Computational Mathematics and Mathematical Physics*, 49(5):797–809, 2009.

[BJ17] Florian Buchegger and Bert Jüttler. Planar multi-patch domain parameterization via patch adjacency graphs. *Computer-Aided Design*, 82:2–12, 2017.

[BJM16] Florian Buchegger, Bert Jüttler, and Angelos Mantzaflaris. Adaptively refined multi-patch B-splines with enhanced smoothness. *Applied Mathematics and Computation*, 272:159–172, 2016.

[BZ09] Lorenz T Biegler and Victor M Zavala. Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575–582, 2009.

[FJ19] Antonella Falini and Bert Jüttler. THB-splines multi-patch parameterization for multiply-connected planar domains via template segmentation. *Journal of Computational and Applied Mathematics*, 349:390–402, 2019.

[GC14] Longfei Gao and Victor M Calo. Fast isogeometric solvers for explicit dynamics. *Computer Methods in Applied Mechanics and Engineering*, 274:19–41, 2014.

[GENN12] Jens Gravesen, Anton Evgrafov, Dang-Manh Nguyen, and Peter Nørtoft. Planar parametrization in isogeometric analysis. In *International Conference on Mathematical Methods for Curves and Surfaces*, pages 189–212. Springer, 2012.

[GH73] William J Gordon and Charles A Hall. Transfinite element methods: blending-function interpolation over arbitrary curved element domains. *Numerische Mathematik*, 21(2):109–129, 1973.

[HMV18a] J Hinz, M Möller, and C Vuik. Elliptic grid generation techniques in the framework of isogeometric analysis applications. *Computer Aided Geometric Design*, 2018.

[HMV18b] Jochen Hinz, Matthias Möller, and Cornelis Vuik. Spline-based parameterization techniques for twin-screw machine geometries. In *IOP Conference Series: Materials Science and Engineering*, volume 425, page 012030. IOP Publishing, 2018.

[HMV19] Jochen Hinz, Matthias Möller, and Cornelis Vuik. An IGA framework for pde-based planar parameterization with arbitrary interface continuity. *arXiv preprint arXiv:1904.03009*, 2019.

[KK04] Dana A Knoll and David E Keyes. Jacobian-free newton–krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[LB07] P Lamby and KH Brakhage. Elliptic grid generation by B-spline collocation. In *Proceedings of the 10th International Conference on Numerical Grid Generation in Computational Field Simulations, FORTH, Crete, Greece*, 2007.

[Man89] J Manke. A tensor product B-spline method for numerical grid generation. Technical report, Washington Univ., Seattle, WA (USA). Dept. of Applied Mathematics, 1989.

[SS89] Mary C Seiler and Fritz A Seiler. Numerical recipes in C: The art of scientific computing. *Risk Analysis*, 9(3):415–416, 1989.

[SY97] Richard M Schoen and Shing-Tung Yau. *Lectures on harmonic maps*, volume 2. Amer Mathematical Society, 1997.

[Win81] Alan M Winslow. Adaptive-mesh zoning by the equipotential method. Technical report, Lawrence Livermore National Lab., CA (USA), 1981.

[XKFC18] Shiwei Xiao, Hongmei Kang, Xiao-Ming Fu, and Falai Chen. Computing IGA-suitable planar parameterizations by polysquare-enhanced domain partition. *Computer Aided Geometric Design*, 62:29–43, 2018.

4

GOAL-ORIENTED ADAPTIVE THB-SPLINE SCHEMES FOR PDE-BASED PLANAR PARAMETERIZATION

The mappings produced by the PDE-based parameterization approach may fail to fulfil the theoretically-predicted bijectivity property due to the truncation error of the numerical scheme. As mentioned in Chapter 2, this may be repaired by refining the underlying spline space. However, determining which regions should be marked for refinement is nontrivial. This may lead to over- or under-refinement and hinders automation, making the incorporation of a posteriori refinement into an automated parameterization framework difficult. This chapter is motivated by our striving for a more robust approach. For this, this chapter employs an unstructured spline technology and determines the region of refinement using duality considerations, avoiding over- and under-refinement. Furthermore, the parametric properties of the outcome are controlled through an appropriate coordinate transformation. While parametric control is discussed in the classical literature [Ste93, TSW98], the combination with a posteriori refinement is a scarce topic. To the best of our knowledge, the only publication that addresses this point can be found in [Aza09], where uniform h-refinement is performed in a classical finite difference setting.

This work considers geometries with complex boundary contours in order to highlight the advantages of local refinement. However, the techniques discussed can also be applied in a structured spline setting. The test cases can hence be considered a robustness *stress test* for the proposed a posteriori refinement strategies.

4.1. INTRODUCTION

Isogeometric analysis (IGA), first introduced by Hughes et al. in [CHB09], is a numerical technique that aims to bridge the gap between computer aided design (CAD) and (isoparametric) finite element analysis (FEA). This is accomplished by building the geometry mapping from the same spline basis that is used to approximately solve PDE-problems posed over the geometry. As such, spline-based parameterization techniques have received an increased amount of interest in the mathematical community in recent years. Since the CAD pipeline typically provides no more than a spline-based description of the boundary contours of the target geometry, the purpose of all parameterization algorithms is to generate a bijective (folding-free) geometry parameterization from the boundary CAD data. Analogous to mesh quality in classical FEA, the parametric quality of the surface parameterization has a profound impact on the numerical accuracy of the isogeometric analysis [XMDG10]. Therefore, besides bijectivity, proficient parameterization algorithms aim at generating parameterizations of high numerical quality.

One of the most important applications of IGA lies in shape optimization problems. Since the geometry changes at every shape optimization iteration, algorithms that are differentiable with respect to the design variables (i.e., the boundary control points) have a further advantage since they allow for employing gradient-based shape optimization algorithms which tend to converge in fewer iterations than their zeroth-order counterparts. Another advantage of differentiability is efficiency: as the inner control points are a smooth function of the boundary control points, there is no need for full remeshing after each iteration since cheaper mesh update strategies can be employed. This is also true for settings in which the boundary contours change as a smooth function of time.

Traditionally, parameterizations for IGA-applications are built from tensor-product spline spaces. Unfortunately, structured spline technologies do not allow for local refinement as knot insertion in one parametric direction automatically refines a whole row / column of the underlying spline space. For the geometry description, this may result in a very dense spline basis whenever many degrees of freedom (DOFs) are required to properly resolve the boundary contours. As a result, the total number of unknowns (the inner control points) may become infeasibly large, leading to a severe slow-down of the meshing process and / or the isogeometric analysis.

To address above efficiency concerns, this chapter introduces a PDE-based planar parameterization framework that uses THB-splines [GJS12], an unstructured spline technology which allows for local refinement, potentially reducing the required total number of DOFs. A major challenge of unstructured spline technologies is deciding where local refinement is required and where a lower resolution suffices. For this, we employ the principles of *dual weighted residual* (DWR) [Ran04], an a posteriori refinement technique for PDE problems based on duality considerations. Furthermore, we augment the problem formulation with a mechanism that allows for changing the parametric properties of the PDE solution in order to fine-tune the parametric properties of the mapping operator.

4.1.1. CHAPTER NOTATION

In this chapter, we denote vectors in boldface. The i-th entry of a vector $\mathbf{x}$ is denoted by $\mathbf{x}_i$ or simply x_i and similarly for the ij-th entry of matrices. We make extensive use of

vector derivatives. Here, we interchangeably use the denotation

$$\partial_{\mathbf{t}}\mathbf{x} \equiv \frac{\partial \mathbf{x}}{\partial \mathbf{t}}, \quad \text{with} \quad \left[\frac{\partial \mathbf{x}}{\partial \mathbf{t}}\right]_{ij} = \frac{\partial x_i}{\partial t_j} \tag{4.1}$$

for the partial derivative.
Furthermore, we frequently work with spline vector spaces $\mathcal{V}_h$. Here, $[\mathcal{V}_h]$ refers to the canonical (THB-)spline basis of $\mathcal{V}_h$, which we assume to be clear from context.

4.1.2. PROBLEM STATEMENT

Let Ω denote the target geometry and $\hat{\Omega} = (0,1)^2$ the parametric domain. In general, we assume that Ω is topologically equivalent to $\hat{\Omega}$. By $\mathbf{x} : \hat{\Omega} \to \Omega$, we denote the mapping operator whose components are built from the linear span of the (THB-)spline basis $[\mathcal{V}_h] = \{w_1, \dots, w_N\}$. The mapping operator $\mathbf{x} : \hat{\Omega} \to \Omega$ is of the form:

$$\mathbf{x}(\xi,\eta) = \sum_{i \in \mathcal{I}_I} \mathbf{c}_i w_i(\xi,\eta) + \sum_{j \in \mathcal{I}_B} \mathbf{c}_j w_j(\xi,\eta), \tag{4.2}$$

where $\mathcal{I}_I$ and $\mathcal{I}_B$ refer to the index-sets corresponding to vanishing and nonvanishing basis functions on $\partial\hat{\Omega}$, respectively and $\mathbf{c}_k \in \mathbb{R}^2, \ \forall k \in \mathcal{I}_I \cup \mathcal{I}_B$. Here, $\mathcal{I}_I$ corresponds to the subspace $\mathcal{V}_h^\circ = \mathcal{V}_h \cap H_0^1(\hat{\Omega})$. Note that $\mathcal{I}_I$ and $\mathcal{I}_B$ are mutually disjoint and

$$\mathcal{I}_I \cup \mathcal{I}_B = \{1, \dots, N\}.$$

In general, we assume that the $\mathbf{c}_j$ in (4.2) are chosen such that $\mathbf{x}|_{\hat{\Omega}}$ is a Jordan curve that parameterizes $\partial\Omega$.
Then, the purpose of any parameterization algorithm is to choose the $\mathbf{c}_i$ in (4.2) such that

1. $\mathbf{x} : \hat{\Omega} \to \Omega$ is bijective,
2. $\mathbf{x}$ is a parameterization of high numerical quality,

while the $\mathbf{c}_j$ are typically held fixed.

4.1.3. RELATED WORK

Existing parameterization techniques can be divided into three broad categories:

1. Algebraic (direct) methods;
2. methods based on (constrained and unconstrained) quality cost function optimization;
3. PDE-based methods.

Algebraic methods (1.) generate a mapping from the solution of a linear system of equations or the evaluation of a closed-form expression. The most-widely used algebraic

method is based on the Coon's patch approach [FH99]. Given the four (known) boundary curves $\mathbf{x}(\xi,0)$, $\mathbf{x}(1,\eta)$, $\mathbf{x}(\xi,1)$ and $\mathbf{x}(0,\eta)$, the mapping is constructed by projecting the components of

$$\begin{aligned}\mathbf{x}_{\text{Coons}} = \;& (1-\xi)\mathbf{x}(0,\eta) + \xi\mathbf{x}(1,\eta) \\ + \;& (1-\eta)\mathbf{x}(\xi,0) + \eta\mathbf{x}(\xi,1) \\ - \;& \begin{bmatrix}1-\xi & \xi\end{bmatrix}\begin{bmatrix}\mathbf{x}(0,0) & \mathbf{x}(0,1)\\ \mathbf{x}(1,0) & \mathbf{x}(1,1)\end{bmatrix}\begin{bmatrix}1-\eta\\ \eta\end{bmatrix}\end{aligned} \tag{4.3}$$

onto the spline space $\mathcal{V}_h$. Whenever $[\mathcal{V}_h]$ is a tensor-product spline basis, the inner control points can also be computationally inexpensively computed from an explicit formula, see [FH99], while in an unstructured setting equation (4.3) can be used.
Another class of algebraic methods results from minimizing a convex, quadratic cost function $Q(\mathbf{x})$ over the inner control points $\mathbf{c}_i, i \in \mathcal{I}_I$. As before, the boundary control points follow from the boundary contours and are held fixed. $Q(\mathbf{x})$ is typically given by a positively-weighted sum of several cost functions. As such, it takes the form:

$$Q(\mathbf{x}) = \sum_i \underbrace{\lambda_i}_{\geq 0} Q_i(\mathbf{x}), \tag{4.4}$$

while the minimization problem becomes:

$$\int_{\hat{\Omega}} Q(\mathbf{x})\mathrm{d}S \to \min_{\mathbf{x}\in\mathcal{V}_h^2}, \quad \text{s.t.} \quad \mathbf{x}|_{\partial\hat{\Omega}} = \partial\Omega. \tag{4.5}$$

Possible choices for the $Q_i(\mathbf{x})$ in (4.4) are [FŠJ15]:

$$Q_{\text{length}}(\mathbf{x}) = \|\mathbf{x}_\xi\|^2 + \|\mathbf{x}_\eta\|^2 \quad \text{and} \quad Q_{\text{uniformity}}(\mathbf{x}) = \|\mathbf{x}_{\xi\xi}\|^2 + 2\|\mathbf{x}_{\xi\eta}\|^2 + \|\mathbf{x}_{\eta\eta}\|^2, \tag{4.6}$$

where the latter requires $\mathcal{V}_h \subset C^1(\hat{\Omega})$. The minimization of (4.5) converges after one iteration of a Newton-type optimization algorithm and can hence be considered of type (1.) as well as type (2.). For an overview of type (1.) approaches, we refer to [GENN12].
Another convex but quartic cost function is the *Liao*-functional [Ste93]

$$Q_{\text{Liao}} = g_{11}^2 + 2g_{12}^2 + g_{22}^2, \tag{4.7}$$

where the g_{ij} denote the entries of the metric tensor of the mapping, with

$$g_{ij} = \mathbf{x}_{\boldsymbol{\xi}_i} \cdot \mathbf{x}_{\boldsymbol{\xi}_j} \quad \text{and} \quad \boldsymbol{\xi} = (\xi_1, \xi_2)^T \equiv (\xi,\eta)^T. \tag{4.8}$$

The minimization of above cost functions is computationally efficient, thanks to convexity, however, the resulting mappings are often folded, i.e., they do not satisfy:

$$\det J > 0, \quad \forall (\xi,\eta)^T \in \hat{\Omega}, \quad \text{where} \quad J(\mathbf{x}) = \partial_{\boldsymbol{\xi}}\mathbf{x} \tag{4.9}$$

denotes the Jacobian of $\mathbf{x}$.
The minimization of nonconvex quality functionals is computationally more demanding but tends to yield better results when convex optimization leads to a folded mapping [Ste93]. Typical nonconvex quality functionals are:

- The *area* functional

$$Q_{\text{area}} = \det J^2, \tag{4.10}$$

  which aims to minimize the variance of $\det J$ over $\hat{\Omega}$;

- the *orthogonality* functional

$$Q_{\text{Orthogonality}} = g_{12}^2 \quad \text{or} \quad Q_{\text{AreaOrthogonality}} = g_{11}\, g_{22}, \tag{4.11}$$

  which is aimed at orthogonalizing the parameter lines;

- the *eccentricity* functional

$$Q_{\text{eccen}} = \left(\frac{\mathbf{x}_\xi \cdot \mathbf{x}_{\xi\xi}}{g_{11}} \right)^2 + \left(\frac{\mathbf{x}_\eta \cdot \mathbf{x}_{\eta\eta}}{g_{22}} \right)^2, \tag{4.12}$$

  which penalizes fast accelerations along the parameter lines.

Unfortunately, minimization of the above functionals, in many cases, leads to folding, too. To the best of our knowledge, there are two main ways to prevent the grid from folding:

(a) Penalization;

(b) constrained minimization.

Option (a) attempts to prevent grid folding through the modification of existing cost functions with a penalty term, such as

- the *Modified Liao* functional

$$Q_{\text{ML}} = \left(\frac{g_{11} + g_{22}}{\det J} \right)^2. \tag{4.13}$$

Adding the Jacobian determinant in the denominator serves the purpose of mitigating the tendency to fold, since the cost functional possesses an infinite barrier close to the boundary of the feasible region.
The most widely-used penalty cost functional is the so-called

- *Winslow* functional

$$Q_{\text{W}} = \frac{g_{11} + g_{22}}{\det J}. \tag{4.14}$$

With $\mathbf{x} \equiv (x, y)^T$, the *Winslow* functional (4.14) follows from performing a pullback of the problem

$$\frac{1}{2} \int_\Omega \left\| \boldsymbol{\xi}_x \right\|^2 + \left\| \boldsymbol{\xi}_y \right\|^2 \mathrm{d}\mathbf{x} \to \min_{\mathbf{x}}, \quad \text{s.t.} \quad \boldsymbol{\xi}|_{\partial\Omega} = \partial\hat{\Omega} \tag{4.15}$$

into $\hat{\Omega}$. An approach based on the *Winslow* functional can be regarded as the *mapping inverse* counterpart of an approach based on the length functional (4.6).
In option (b), the minimization is carried out with an added constraint that constitutes a sufficient condition for (4.9). For tensor-product B-spline bases, in [XMDG11], Xu et al. propose a linear convex sufficient condition $L(\mathbf{x}) > 0$ for bijectivity. It is added as a constraint to the minimization problem. If convex cost functions are utilized, this leads to a linear programming problem, which can be computationally inexpensively solved using convex optimization routines. Unfortunately, the set

$$\left\{\mathbf{x} \in \mathcal{V}_h^2 \;\middle|\; \mathbf{x}|_{\partial\hat{\Omega}} = \partial\Omega \text{ and } L(\mathbf{x}) > 0\right\}$$

may be empty or the constraint may be very restrictive, limiting its applicability to relatively simple shapes.
In an effort to allow for more complicated shapes, [XMDG11] and [GENN12] propose nonlinear nonconvex sufficient conditions for bijectivity. Since the Jacobian determinant $\det J$ is a piecewise-polynomial function of higher polynomial degree itself, it can be projected onto a spline basis that contains it. If all the weights are positive under the expansion, this constitutes a sufficient condition for bijectivity. The nonlinear sufficient condition $N(\mathbf{x}) > 0$ is added as a constraint and the optimization is carried out with a blackbox nonlinear optimization routine (typically, IPOPT [BZ09]) that comes with all the drawbacks of nonconvex optimization such as the danger of getting stuck in local minima. A further disadvantage is the need for an initial guess that satisfies the constraints, for which another nonconvex optimization problem has to be solved first.
While the extension of (penalized or unpenalized) cost function minimization to THB-splines is straightforward, this is not the case for constrained methods, since the constraints are designed for structured splines only. To the best of our knowledge, the only comprehensive overview of planar parameterization techniques for THB-splines can be found in [FŠJ15], where the application of most of the mentioned (unpenalized) cost functions is studied in a THB-setting. As the optimization is carried out without constraints, folding occurs in the majority of test cases. The paper concludes that the only method potentially capable of dealing with arbitrarily-complex shapes is based on computing $\mathbf{x}$ by approximating the inverse of a map $\mathbf{h}$ which is comprised of a pair of harmonic functions in Ω, i.e.,

$$\Delta\mathbf{h} = \mathbf{0} \quad \text{in } \Omega, \quad \text{s.t.} \quad \mathbf{h}|_{\partial\Omega} = \partial\hat{\Omega}. \tag{4.16}$$

The authors propose a two-step approach: First a large number of tuples $\left(\mathbf{h}(\mathbf{x}_j), \mathbf{x}_j\right)$, with $\mathbf{x}_j \in \Omega$ is computed using an isogeometric boundary element method [RS07, SS10], after which the pairs are utilized to approximate $\mathbf{h}^{-1}$ through a least-squares minimization problem with regularization terms.
Above methodology is equivalent to minimizing the *Winslow* functional (4.14), which follows straightforwardly from deriving the Euler-Lagrange equations of the minimization problem (4.15). As $\mathbf{h}$ is a pair of harmonic functions with convex target domain, it follows from the Radó-Kneser-Choquet theorem that $\mathbf{h}$ is a diffeomorphism in the interior of Ω [GENN12], justifying an approximation of its inverse for the purpose of computing a domain parameterization.
A major advantage of the two-step approach from [FŠJ15] over a direct minimization of

(4.14) is that the latter requires a folding-free initial domain parameterization to avoid division by zero. In the vast majority of cases, however, such a bijection is not available, limiting the method's scope to improving the parametric properties of an already bijective mapping.
An advantage of minimizing (4.14), however, is that if the global minimum over $\mathcal{V}_h^2$ has been found, it is clearly bijective, while bijectivity may be lost in the indirect approach, due to numerical inaccuracies.
The observation that the impractical minimization of the *Winslow* functional (4.14) is equivalent to solving an inverse Laplace problem has lead to the development of (3.) PDE-based parameterization methods. To acquire a PDE-problem posed over $\hat{\Omega}$, we perform a pullback:

$$\Delta_{\mathbf{x}}\boldsymbol{\xi} = \mathbf{0} \quad \text{in } \hat{\Omega}, \quad \text{s.t.} \quad \mathbf{x}|_{\partial\hat{\Omega}} = \partial\Omega, \tag{4.17}$$

where $\Delta_{\mathbf{x}}$ denotes the Laplace-Beltrami [Kre91] operator with respect to $\mathbf{x}$. Problem (4.17) suffers from the same shortcoming of the *Winslow*-approach: the appearance of a Jacobian determinant in the denominator. However, we may scale the equation by multiplying with any nonsingular 2×2 tensor T. Choosing $T = (\det\mathbf{x}_{\boldsymbol{\xi}})^2\mathbf{x}_{\boldsymbol{\xi}}$ (which is nonsingular thanks to the theoretically predicted bijectivity of the PDE-solution), the Jacobian determinant can be removed from (4.17), leading to the following quasi-linear second-order PDE problem [HMV18b]:

$$A(\mathbf{x})\colon H(\mathbf{x}_i) = 0 \quad \text{in } \hat{\Omega}, \quad \text{for } i \in \{1,2\} \quad \text{s.t.} \quad \mathbf{x}|_{\partial\hat{\Omega}} = \partial\Omega, \tag{4.18}$$

where

$$H(u)_{ij} = \frac{\partial^2 u}{\partial\boldsymbol{\xi}_i\partial\boldsymbol{\xi}_j} \quad \text{and} \quad A(\mathbf{x}) = \frac{1}{g_{11}+g_{22}+\epsilon}\begin{pmatrix} g_{22} & -g_{12} \\ -g_{12} & g_{11}\end{pmatrix}, \tag{4.19}$$

with the g_{ij} as in (4.8) and ϵ a small positive constant that serves numerical stability (typically, $\epsilon \simeq 10^{-4}$). Furthermore, $A\colon B$ denotes the Frobenius inner product.

Remark. The purpose of dividing by $g_{11}+g_{22}+\epsilon$ in (4.19) is achieving scaling invariance.

In [HMV18a], equation (4.18) is discretized with a Galerkin approach. The resulting equations are tackled with a Newton-based iterative approach, which is initialized with an algebraic initial guess.
The advantages and disadvantages of solving (4.18) over a direct minimization of (4.14) are the same as in the indirect approach from [FŠJ15]. Hence, folding resulting from insufficient numerical accuracy can be resolved by recomputing the mapping from a refined spline space.
In this chapter, we will present several schemes for approximately solving (4.18) with THB-spline bases. A major challenge in a THB-setting is deciding where a high resolution is needed. Since the approach is PDE-based, we adopt the a posteriori refinement strategy of *dual weighted residual*, which is the topic of Section 4.3.1.

4.2. SOLUTION STRATEGIES

In this section we present several solution strategies to approximately solve (4.18). As the resulting equations are nonlinear, we base our solution strategy on iterative approaches.

Initial guesses are always constructed using an algebraic method (see Section 4.1).
Let

$$\mathscr{U}^{\mathbf{f}} = \left\{ \mathbf{v} \in \mathcal{V}^2 \;\middle|\; \mathbf{v} = \mathbf{f} \text{ on } \partial\hat{\Omega} \right\} \tag{4.20}$$

and let $\mathscr{U}_h^{\mathbf{f}}$ be the set resulting from replacing $\mathcal{V}$ by the finite-dimensional $\mathcal{V}_h \subset \mathcal{V}$ in (4.20). We have

$$\mathscr{U}_h^{\mathbf{f}} = \left\{ \mathbf{v} \in \mathcal{V}_h^2 \;\middle|\; \mathbf{v} = \mathbf{f} \text{ on } \partial\hat{\Omega} \right\}. \tag{4.21}$$

Remark. For $\mathscr{U}_h^{\mathbf{f}}$ in (4.21) to be nonempty, we have to assume that $\mathbf{f}$ restricted to $\partial\hat{\Omega}$ is contained in $\mathcal{V}_h^2$, which may necessitate a projection of the Dirichlet data onto the finite-dimensional (THB-)spline space $\mathcal{V}_h^2$.

Let $\mathbf{x}_D$ be such that $\mathbf{x}_D|_{\partial\hat{\Omega}}$ parameterizes $\partial\Omega$. For convenience we assume that $\mathbf{x}_D \in \mathcal{V}_h^2 \setminus \mathscr{U}_h^{\mathbf{0}}$. In an IGA-setting, (4.18) suggests a discretization of the form:

$$\text{find } \mathbf{x}_h \in \mathscr{U}_h^{\mathbf{x}_D} \quad \text{s.t.} \quad F(\mathbf{x}_h, \boldsymbol{\sigma}_h) = 0 \quad \forall \boldsymbol{\sigma}_h \in \mathscr{U}_h^{\mathbf{0}}, \tag{4.22}$$

with

$$F(\mathbf{x}, \boldsymbol{\sigma}) = \sum_{i=1}^{2} \int_{\hat{\Omega}} \boldsymbol{\tau}_i(\boldsymbol{\sigma}, \mathbf{x}) A(\mathbf{x}) : H(\mathbf{x}_i) \mathrm{d}S, \tag{4.23}$$

for some $\boldsymbol{\tau} : \mathscr{U}^{\mathbf{0}} \times \mathcal{V}^2 \to L_2(\hat{\Omega}, \mathbb{R}^2)$. Unless stated otherwise, in the following, we assume $\boldsymbol{\tau}(\boldsymbol{\sigma}, \mathbf{x}) = \boldsymbol{\sigma}$.
As second order derivatives appear in (4.23), in (4.20) we take $\mathcal{V} = H^2(\hat{\Omega})$.

4.2.1. NEWTON APPROACH

In the following, we briefly recapitulate the approach from [HMV18a], which is designated for tensor-product NURBS bases but can also be applied in a THB-setting. By

$$B'(u, \ldots, z) \equiv \left. \frac{\partial B(u + \epsilon z, \ldots)}{\partial \epsilon} \right|_{\epsilon = 0}, \tag{4.24}$$

we denote the Gateaux derivative of any differentiable form $B(\cdot, \ldots)$ with respect to its first argument. Given $\mathbf{x}^k \in \mathscr{U}_h^{\mathbf{x}_D}$, we compute the Newton increment from

$$\text{find } \delta\mathbf{x}^k \in \mathscr{U}_h^{\mathbf{0}} \quad \text{s.t.} \quad F'(\mathbf{x}^k, \boldsymbol{\sigma}_h, \delta\mathbf{x}^k) = -F(\mathbf{x}^k, \boldsymbol{\sigma}_h), \quad \forall \boldsymbol{\sigma}_h \in \mathscr{U}_h^{\mathbf{0}}. \tag{4.25}$$

Upon completion, we update $\mathbf{x}^{k+1} = \mathbf{x}^k + \kappa\delta\mathbf{x}^k$ for some $\kappa \in (0, 1]$, whose optimal value is estimated using a line search routine. Above steps are repeated until the residual norm is deemed sufficiently small.
Optionally, derivative evaluations of the form $F'(\mathbf{x}^k, \boldsymbol{\sigma}_h, \mathbf{v})$ may be approximated using finite differences:

$$F'(\mathbf{x}^k, \boldsymbol{\sigma}_h, \mathbf{v}) \simeq \frac{F(\mathbf{x}^k + \epsilon\mathbf{v}, \boldsymbol{\sigma}_h) - F(\mathbf{x}^k, \boldsymbol{\sigma}_h)}{\epsilon}, \tag{4.26}$$

for ϵ small. Solving (4.25) using a suitable Krylov-subspace method only requires computing derivative evaluations $F'(\mathbf{x}^k, \boldsymbol{\sigma}_h, \mathbf{v})$, which may be approximated using (4.26), leading to a Newton-Krylov algorithm that avoids the expensive assembly of the Jacobian matrix in (4.25). The optimal choice of ϵ in (4.26) is discussed in [KK04].

4.2.2. PSEUDO-TRANSIENT CONTINUATION

In this technique, we seek the steady-state solution of the problem

$$\text{find } \mathbf{x}_h(\boldsymbol{\xi},t) \in \mathscr{U}_h^{\mathbf{x}_D}, \quad \text{s.t.} \quad \langle \partial_t \mathbf{x}_h, \boldsymbol{\sigma}_h \rangle = -F(\mathbf{x}_h, \boldsymbol{\sigma}_h), \quad \forall \boldsymbol{\sigma}_h \in \mathscr{U}_h^{\mathbf{0}}, \tag{4.27}$$

with

$$\langle \partial_t \mathbf{x}_h, \boldsymbol{\sigma}_h \rangle = \int_{\hat{\Omega}} \boldsymbol{\sigma}_h \cdot \partial_t \mathbf{x}_h \mathrm{d}S. \tag{4.28}$$

Here, we only consider the choice $\boldsymbol{\tau}(\boldsymbol{\sigma},\mathbf{x}) = \boldsymbol{\sigma}$. We discretize in time using backward Euler. Introducing $\delta\mathbf{x}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$, with $F(\mathbf{x}^{k+1}, \boldsymbol{\sigma}_h) \simeq F(\mathbf{x}^k, \boldsymbol{\sigma}_h) + F'(\mathbf{x}^k, \boldsymbol{\sigma}_h, \delta\mathbf{x}^k)$, we compute the temporal increment from

$$\text{find } \delta\mathbf{x}^k \in \mathscr{U}_h^{\mathbf{0}}, \quad \text{s.t.} \quad \left\langle \frac{\delta\mathbf{x}^k}{\delta t^k}, \boldsymbol{\sigma}_h \right\rangle + F'(\mathbf{x}^k, \boldsymbol{\sigma}_h, \delta\mathbf{x}^k) = -F(\mathbf{x}^k, \boldsymbol{\sigma}_h), \quad \forall \boldsymbol{\sigma}_h \in \mathscr{U}_h^{\mathbf{0}}, \tag{4.29}$$

where δt^k denotes the time-step during the k-th iteration. As proposed in [KK98], we base the time-step selection on the following recursive formula

$$\delta t^k = \delta t^{k-1} \frac{\left\|\mathbf{F}(\mathbf{x}^{k-1})\right\|_2}{\left\|\mathbf{F}(\mathbf{x}^k)\right\|_2}, \quad \text{with} \quad \|\mathbf{F}(\mathbf{x})\|_2^2 = \sum_{\boldsymbol{\sigma}_h \in \left[\mathscr{U}_h^{\mathbf{0}}\right]} F(\mathbf{x}, \boldsymbol{\sigma}_h)^2. \tag{4.30}$$

The iteration is terminated once $\|\mathbf{x}^k - \mathbf{x}^{k-1}\|$ is sufficiently small (in a suitable norm).

4.2.3. PICARD ITERATION

In the following, we present a Picard-based iterative scheme that is loosely based on the default approach from the rich literature of classical meshing techniques [TSW98]. As opposed to Sections 4.2.1 and 4.2.2, we base the scheme on a linearize then discretize approach, rather than the converse. Note that for given $\mathbf{x} = (x,y)^T$, we have

$$A(\mathbf{x}) = C^T(\mathbf{x})C(\mathbf{x}), \quad \text{with} \quad C(\mathbf{x}) = \frac{1}{\sqrt{g_{11} + g_{22} + \epsilon}} \begin{pmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial y}{\partial \xi} \\ -\frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \end{pmatrix}. \tag{4.31}$$

As such, $A(\mathbf{x})$ is symmetric positive semi-definite (SPSD) for all $\mathbf{x}$ and symmetric positive definite (SPD) for $\mathbf{x}: \hat{\Omega} \to \Omega$ bijective. Let us introduce the operator $\mathbf{K}: C^2(\hat{\Omega},\mathbb{R}^2) \times C^2(\hat{\Omega},\mathbb{R}^2) \times \mathbb{R}^+ \to C^0(\hat{\Omega},\mathbb{R}^2)$ with components

$$\mathbf{K}_i(\mathbf{x},\mathbf{y},\mu) = A_\mu(\mathbf{y}) : H(\mathbf{x}_i) - \mu\Delta_{\boldsymbol{\xi}}\mathbf{y}_i, \quad \text{where} \quad A_\mu(\mathbf{y}) = A(\mathbf{y}) + \mu I^{2\times 2}. \tag{4.32}$$

Note that for $\mu > 0$, $A_\mu(\mathbf{x})$ is SPD and that for all choices of μ, $\mathbf{K}_i(\mathbf{x},\mathbf{x},\mu) = A(\mathbf{x}) : H(\mathbf{x}_i)$. For given $\mu > 0$, we seek $\mathbf{x}$ as the limit $k \to \infty$ of the recursive sequence

$$\text{find } \mathbf{x}^{k+1} \quad \text{s.t.} \quad \mathbf{K}(\mathbf{x}^{k+1}, \mathbf{x}^k, \mu) = \mathbf{0}, \quad \text{and} \quad \mathbf{x}^{k+1} = \mathbf{x}_D \quad \text{on} \quad \partial\hat{\Omega}. \tag{4.33}$$

To discretize (4.33), let us introduce the semi-linear form $G_\tau : \mathscr{V}^2 \times \mathscr{V}^2 \times \mathbb{R}^+ \times \mathscr{U}^{\mathbf{0}} \to \mathbb{R}$ with

$$G_\tau(\mathbf{x},\mathbf{y},\mu,\boldsymbol{\sigma}) = \sum_{i=1}^{2} \int_{\hat{\Omega}} \boldsymbol{\tau}_i(\boldsymbol{\sigma},\mathbf{y}) \left(A_\mu(\mathbf{y}) : H(\mathbf{x}_i) - \mu\Delta_{\boldsymbol{\xi}}\mathbf{y}_i \right) \mathrm{d}S. \tag{4.34}$$

Given $\mathbf{x}^k$, we compute $\mathbf{x}^{k+1} \in \mathscr{U}^{\mathbf{x}_D}$ as the solution of

$$\text{find } \mathbf{x}^{k+1} \in \mathscr{U}^{\mathbf{x}_D} \quad \text{s.t} \quad G_\tau(\mathbf{x}^{k+1}, \mathbf{x}^k, \mu, \boldsymbol{\sigma}_h) = 0, \quad \forall \boldsymbol{\sigma}_h \in \mathscr{U}^{\mathbf{0}}, \tag{4.35}$$

where, as before, $\mathscr{U}^{\mathbf{x}_D} = \{\mathbf{v} \in \mathscr{V}^2 \mid \mathbf{v} = \mathbf{x}_D \text{ on } \partial\hat{\Omega}\}$.
The discretization of (4.35) follows straightforwardly from replacing $\mathscr{V}$ by the finite dimensional $\mathscr{V}_h \subset \mathscr{V}$. Equation (4.35) leads to a decoupled (block-diagonal) system of elliptic equations in nonvariational (or non-divergence) form [LP11]. Inspired by [Gal17], here we consider the choices

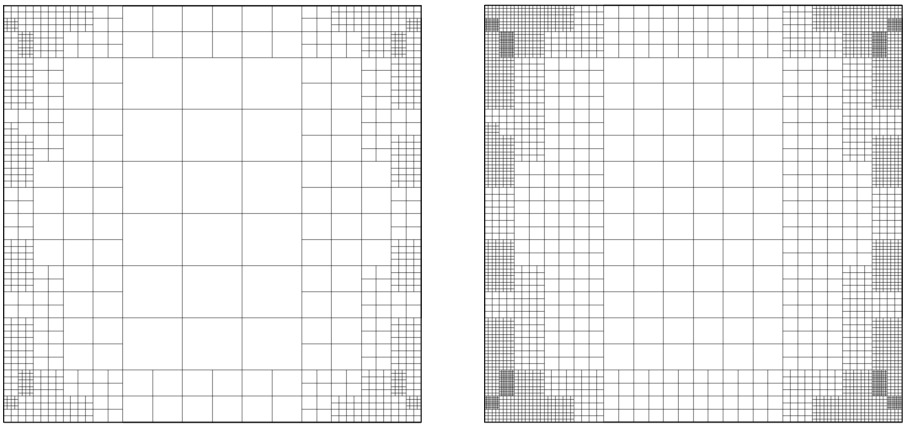
$$\boldsymbol{\tau}^{\mathrm{Id}}(\boldsymbol{\sigma}, \mathbf{y}) = \boldsymbol{\sigma}, \quad \boldsymbol{\tau}^{\mathrm{div}}(\boldsymbol{\sigma}, \mathbf{y}) = \gamma(\mathbf{y}) \Delta_{\boldsymbol{\xi}} \boldsymbol{\sigma} \quad \text{and} \quad \boldsymbol{\tau}_i^{\mathrm{ls}}(\boldsymbol{\sigma}, \mathbf{y}) = A_\mu(\mathbf{y}) : H(\boldsymbol{\sigma}_i), \tag{4.36}$$

where

$$\gamma(\mathbf{y}) = \frac{\operatorname{trace}(A_\mu(\mathbf{y}))}{A_\mu : A_\mu(\mathbf{y})}. \tag{4.37}$$

A Picard scheme results from iterating on (4.35) until $\|\mathbf{x}^{k+1} - \mathbf{x}^k\|$ is negligibly small.

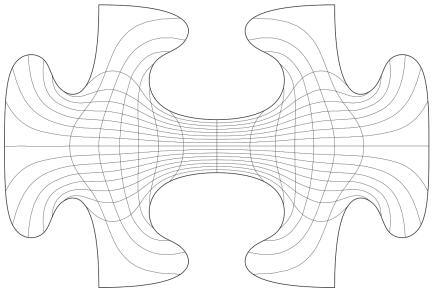
Remark. Adding artificial diffusion in (4.32) stabilizes the linearized discrete equation from (4.35). In the absence of stabilization (i.e., $\mu = 0$), (4.35) can be ill-posed in rare cases, depending on the previous iterate $\mathbf{x}^k$. This is also true for a Newton-based approach. Whenever an invalid iterate is encountered in the Newton approach, we fall back on the techniques from this section.
For $\mu > 0$, well-posedness of (4.35) with the choices from (4.36) is discussed in [Gal17] and [BHW19]. Stabilizing a Newton-based approach constitutes a topic for future research.



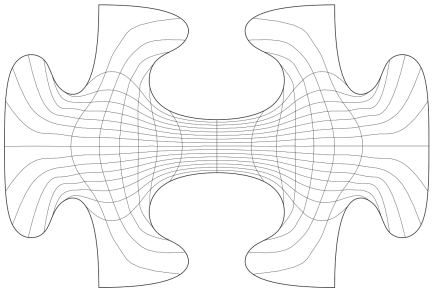
(a) The unrefined domain.

(b) The uniformly refined domain.

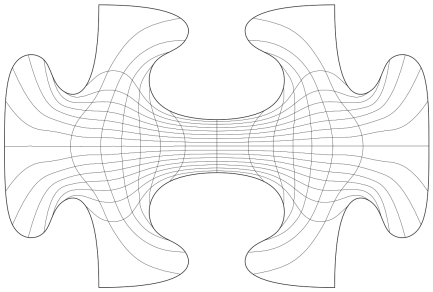
Figure 4.1: The THB-refined parametric domains used in the computations of the parameterizations from Figures 4.2 and 4.3.



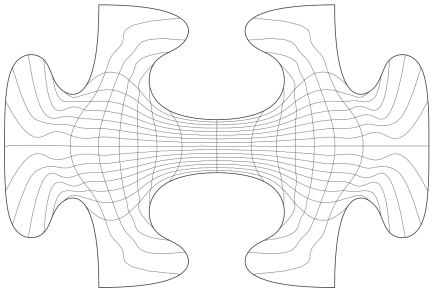
(a) Reference mapping acquired from minimizing the Winslow functional over the domain from Figure 4.1a.



(b) The parameterization for the choice $\boldsymbol{\tau} = \boldsymbol{\tau}^{\mathrm{Id}}$.



(c) The parameterization for the choice $\boldsymbol{\tau} = \boldsymbol{\tau}^{\mathrm{ls}}$.



(d) The parameterization for the choice $\boldsymbol{\tau} = \boldsymbol{\tau}^{\mathrm{div}}$.

Figure 4.2: Parameterizations acquired using the various discretization techniques.

refinement \ method	Direct	$\boldsymbol{\tau} = \boldsymbol{\tau}^{\mathrm{Id}}$	$\boldsymbol{\tau} = \boldsymbol{\tau}^{\mathrm{ls}}$	$\boldsymbol{\tau} = \boldsymbol{\tau}^{\mathrm{div}}$
h	4.784	4.849	4.913	4.974
$h/2$		4.787	4.790	4.815

Table 4.1: Evaluation of the Winslow functional with the various parameterizations.

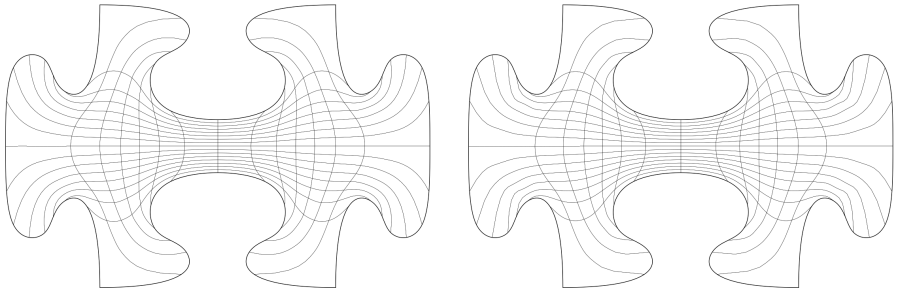
4.2.4. DIRECT APPROACH

Assuming a bijective initial guess $\mathbf{x}^0 \in \mathscr{U}_h^{\mathbf{x}_D}$ is available, we may alternatively compute an approximately inversely harmonic parameterization by a direct minimization of the Winslow functional (4.14). Let

$$L_W(\mathbf{x}) = \int_{\hat{\Omega}} Q_{\mathrm{W}}(\mathbf{x}) \mathrm{d}S \tag{4.38}$$

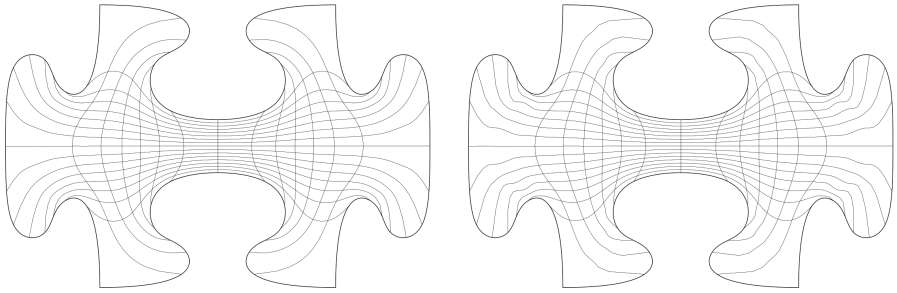
denote the evaluation of the Winslow function (see equation (4.14)), whose domain is the set of all bijective $\mathbf{x}$. To conform with the topic of this chapter, we compute the minimizer over $\mathscr{U}_h^{\mathbf{x}_D}$ as the solution of the following discretized PDE problem:

$$\text{find } \mathbf{x}_h \in \mathscr{U}_h^{\mathbf{x}_D} \quad \text{s.t.} \quad L'_W(\mathbf{x}_h, \boldsymbol{\sigma}_h) = 0, \quad \forall \boldsymbol{\sigma}_h \in \mathscr{U}_h^{\mathbf{0}}. \tag{4.39}$$



(a) Winslow.

(b) The refined parameterization for $\boldsymbol{\tau} = \boldsymbol{\tau}^{\mathrm{Id}}$.



(c) The refined parameterization for $\boldsymbol{\tau} = \boldsymbol{\tau}^{\mathrm{ls}}$.

(d) The refined parameterization for $\boldsymbol{\tau} = \boldsymbol{\tau}^{\mathrm{div}}$.

Figure 4.3: Parameterizations acquired using the various discretization techniques over the uniformly refined domain.

We solve (4.39) with one of the approaches from Sections 4.2.1 and 4.2.2. Typically $\mathbf{x}^0$ is the solution of one of the indirect methods presented in Sections 4.2.1 to 4.2.3. In practice, we have often encountered convergence failure even when $\mathbf{x}^0$ is bijective. As a rule of thumb, we retry solving (4.39) with a refined $\mathbf{x}^0$, resulting from an indirect approach, if converge is not reached after a few iterations.

Remark. If a measure of quality of the solution results from substituting into (4.38), a direct approach yields the best outcome.

4.2.5. EXAMPLE: PUZZLE PIECE

Figure 4.2 shows the various parameterizations of a puzzle piece geometry, resulting from solving the discretized equations with the Newton-approach from Section 4.2.1 and the different choices of $\boldsymbol{\tau} : \mathscr{U}^{\mathbf{0}} \times \mathscr{V}^2 \to L_2(\hat{\Omega}, \mathbb{R}^2)$ from (4.36). For Newton, stabilization is avoided, i.e., $\mu = 0$. All methods lead to a bijective outcome. However, the figure shows noticeable differences in the parametric properties between the various methods, in particular in the protruded parts and in particular in Figure 4.2d. Upon uniform refinement, the differences become less pronounced, suggesting that all schemes are consistent. Table 4.1 shows the outcomes of substituting the various parameterizations into (4.38). Not surprisingly, the choice $\boldsymbol{\tau} = \boldsymbol{\tau}^{\mathrm{div}}$ fares the worst while the table suggests that $\boldsymbol{\tau} = \boldsymbol{\tau}^{Id}$

is the best choice. Upon refinement, the $\boldsymbol{\tau} = \boldsymbol{\tau}^{Id}$ and $\boldsymbol{\tau} = \boldsymbol{\tau}^{ls}$ parameterizations become virtually indistinguishable from the global minimizer over the coarse space, which is also reflected in table 4.1.
As documented in the literature [Ste93], all parameterizations suffer from the well-known pathologies of inversely harmonic maps, such as the tendency to yield large elements within protruded parts. Fortunately, in a THB-setting this can be compensated for by performing local refinement in the affected regions. Mitigating the impact of these pathologies will be the topic of Section 4.4.

4.3. A BASIC SCHEME BASED ON A POSTERIORI REFINEMENT

One of the main challenges of PDE-based parameterization is selecting an appropriate finite-dimensional spline space $\mathcal{V}_h$. For this, we employ the technique of *Dual Weighted Residual*, which will be the topic of Section 4.3.1.

4.3.1. DUAL WEIGHTED RESIDUAL

Dual Weighted Residual, is an a posteriori refinement strategy that is based on duality considerations. Consider a semi-linear differential form $A(u,\phi)$ (which is linear in ϕ). We consider the problem

$$\text{find } u \in \mathcal{V}^\circ \quad \text{s.t.} \quad A(u,\phi) = f(\phi), \quad \forall \phi \in \mathcal{V}^\circ, \tag{4.40}$$

for some linear functional $f(\cdot)$ and a suitably-chosen vector space $\mathcal{V}$, with $\mathcal{V}^\circ = \mathcal{V} \cap H_0^1(\hat{\Omega})$. We seek an approximate solution $u_h \in \mathcal{V}_h^\circ$ with $\mathcal{V}_h \subset \mathcal{V}$ by solving a discretized counterpart of (4.40)

$$\text{find } u_h \in \mathcal{V}_h^\circ \quad \text{s.t.} \quad A(u_h,\phi_h) = f(\phi_h), \quad \forall \phi_h \in \mathcal{V}_h^\circ. \tag{4.41}$$

Let $L(u)$ be such that

$$\Delta L(u_h) \equiv L(u) - L(u_h) \tag{4.42}$$

is a quantity of interest (which for instance measures the global quality of the approximation). Furthermore, let

$$\rho(u,\psi) = f(\psi) - A(u,\psi) \tag{4.43}$$

denote the residual.
If z is the solution of

$$\text{find } z \in \mathcal{V}^\circ \quad \text{s.t.} \quad A'(u,\phi,z) = L'(u,\phi), \quad \forall \phi \in \mathcal{V}^\circ, \tag{4.44}$$

we have

$$\Delta L(u_h) = \rho(u_h, z - \psi_h) + R_h(e), \tag{4.45}$$

for arbitrary $\psi_h \in \mathcal{V}_h^\circ$ and some R_h that is quadratic in $e \equiv u - u_h$ [Ran04]. In practice, we neglect R_h and approximate z by the solution of the discrete adjoint equation

$$\text{find } z_h \in \bar{\mathcal{V}}_h^\circ \quad \text{s.t.} \quad A'(u_h,\sigma_h,z_h) = L'(u_h,\sigma_h), \quad \forall \sigma_h \in \bar{\mathcal{V}}_h^\circ, \tag{4.46}$$

for some adjoint (THB-)spline space $\bar{\mathcal{V}}_h \subset \mathcal{V}$. Hence,

$$\Delta L(u_h) \simeq \rho(u_h, z_h - \psi_h) = \sum_{w_i \in [\mathcal{V}_h]} \rho(u_h, w_i(z_h - \psi_h)) \equiv \sum_i \mathbf{r}_i(u_h), \tag{4.47}$$

thanks to semi-linearity of $A(\cdot,\cdot)$ and the partition of unity property associated with $[\mathcal{V}_h]$. The motivation to use an adjoint spline space that differs from $\mathcal{V}_h$ is the fact that substituting any $z_h \in \mathcal{V}_h^\circ$ in (4.47) results in $\Delta L(u_h) = 0$, making it a lousy approximation due to Galerkin orthogonality.
The appeal of using (4.47) is that a scalar quantity of interest $\Delta L(u_h)$ is transformed into an integral quantity over $\hat{\Omega}$, which in turn is decomposed into the basis function wise contributions $\mathbf{r}_i(u_h)$. The vector $\mathbf{r}(u_h)$ may then be utilized in the selection of basis functions for goal-oriented refinement (see Section 4.3.4).

Remark. If u_h is a very inaccurate approximation of u, the discrete adjoint solution z_h will be inaccurate regardless of the choice of $\bar{\mathcal{V}}_h$. Heuristically, we have rarely encountered this situation in the examples considered in this work. In case refinement is ineffective, the procedure should be restarted with a uniformly refined initial basis.

4.3.2. APPLICATIONS TO PDE-BASED PARAMETERIZATION

In this section, we apply the methodology from Section 4.3.1 to the PDE-based parameterization problem (4.22). Let $\mathbf{x}_D$ be the canonical extension of the Dirichlet data as introduced in (4.22). With $\mathbf{x}_h = \mathbf{x}_D + \mathbf{x}_0$, we may write (4.22) in the equivalent form

$$\text{find } \mathbf{x}_0 \in \mathcal{U}_h^{\mathbf{0}} \quad \text{s.t.} \quad F(\mathbf{x}_D + \mathbf{x}_0, \boldsymbol{\sigma}_h) = 0, \quad \forall \boldsymbol{\sigma}_h \in \mathcal{U}_h^{\mathbf{0}}. \tag{4.48}$$

In the formalism of (4.41), we hence have $A(\mathbf{x}, \boldsymbol{\sigma}) = F(\mathbf{x}_D + \mathbf{x}, \boldsymbol{\sigma})$ and $f(\boldsymbol{\sigma}) = 0$. Alternatively, we may absorb the dependence on $\mathbf{x}_D$ in $f(\cdot)$. As before, the relation between $\mathcal{V}_h$ and $\mathcal{U}_h^{\mathbf{f}}$ follows from (4.21).
We would like to design scalar cost functions ($L(u)$ in (4.42)) to aid us in refining an a priori chosen basis $[\mathcal{V}_h]$ such that after recomputing the solution over the refined space $\mathcal{V}_h^R \supset \mathcal{V}_h$,

1. $\mathbf{x}_h^R$ is bijective;
2. $\mathbf{x}_h^R$ approximates $\mathbf{x}$ well.

In a discrete setting, we may relax the condition that $\mathbf{x}_h^R$ be bijective by the condition that $\mathbf{x}_h^R$ has a positive Jacobian determinant in all quadrature points $\Xi = \{\boldsymbol{\xi}_1^q, \dots, \boldsymbol{\xi}_M^q\}$.
As such, let $\mathbf{x}_h$ be the solution of (4.48) over the space $\mathcal{V}_h$ and let

$$\Xi_- = \left\{ \boldsymbol{\xi}_i^q \in \Xi \mid \det J(\mathbf{x}_h) < 0 \text{ in } \boldsymbol{\xi}_i^q \right\}. \tag{4.49}$$

To address (potential) lack of bijectivity, we propose the following goal-oriented cost function:

$$L_\Xi(\mathbf{x}) = \sum_{\boldsymbol{\xi}_i^q \in \Xi_-} \det J(\mathbf{x})(\boldsymbol{\xi}_i^q), \tag{4.50}$$

such that

$$\Delta L_\Xi(\mathbf{x}_h) = \underbrace{L_\Xi(\mathbf{x})}_{\geq 0} - \underbrace{L_\Xi(\mathbf{x}_h)}_{\leq 0} \geq 0, \tag{4.51}$$

with equality if and only if $\Xi_- = \emptyset$. Here, the inequality $L_\Xi(\mathbf{x}) \geq 0$ follows from the Radó-Kneser-Choquet theorem (see Section 4.1) while $L_\Xi(\mathbf{x}_h) \leq 0$ follows from (4.49). According to (4.47), we may approximate

$$\Delta L_\Xi(\mathbf{x}_h) \simeq -F(\mathbf{x}_h, \mathbf{z}_h - \boldsymbol{\psi}_h) = \sum_{w_i \in [\mathcal{V}_h]} -F\big(\mathbf{x}_h, w_i(\mathbf{z}_h - \boldsymbol{\psi}_h)\big) \equiv \sum_i \mathbf{r}_i(\mathbf{x}_h). \tag{4.52}$$

Typically, we choose $\boldsymbol{\psi}_h$ as the $L_2(\hat{\Omega}, \mathbb{R}^2)$-projection of $\mathbf{z}_h$ onto $\mathcal{U}_h^{\mathbf{0}}$.

Remark. Even though subtracting a nonzero $\boldsymbol{\psi}_h \in \mathcal{U}_h^{\mathbf{0}}$ does not alter the outcome on the right hand side of (4.52), it does influence its decomposition into the basis function wise contributions $\mathbf{r}_i(\mathbf{x}_h)$.

Using the basis function wise decomposition of the quantity of interest $\Delta L_\Xi(\mathbf{x}_h)$, the procedure selects a subset of the $w_i \in [\mathcal{V}_h]$ and marks them for refinement. We propose selection criteria in Section 4.3.4.
After refinement of $\mathcal{V}_h$, we recompute the mapping from the enriched basis $\mathcal{V}_h^R$ and if necessary repeat above steps until discrete bijectivity (over Ξ) has been achieved.

Remark. For better performance, we always use the prolonged coarse-grid solution as an initial guess for recomputing the mapping under the refined basis.

Upon completion, we may choose to settle for the (possibly inaccurate but with respect to the $\boldsymbol{\xi}_i^q \in \Xi$ analysis-suitable) resulting mapping $\mathbf{x}_h^R$, or we may choose to further improve its accuracy with respect to the exact solution. As the exact solution of the PDE problem is equal to the minimizer of the Winslow function (see Section 4.1.3)

$$L_W(\mathbf{x}) = \int_{\hat{\Omega}} \frac{g_{11} + g_{22}}{\det J} \mathrm{d}S,$$

by choosing $-L_W(\mathbf{x})$ as a cost function, we acquire the quantity of interest

$$\Delta L_W(\mathbf{x}_h) = -L_W(\mathbf{x}) + L_W(\mathbf{x}_h) \geq 0, \tag{4.53}$$

with equality for $\|\mathbf{x} - \mathbf{x}_h\|_{H^1(\hat{\Omega})} = 0$. As such, (4.53) may serve as a measure for the distance of $\mathbf{x}_h$ to $\mathbf{x}$. As before, we approximate (4.53) by substituting the discrete adjoint solution $\mathbf{z}_h$ in (4.47) and base refinement criteria on the basis function wise contributions to (4.53). The steps of refinement, recomputation and adjoint estimation may be repeated until the estimate $|\Delta L_W(\mathbf{x}_h)| \simeq |-F(\mathbf{x}_h, \mathbf{z}_h - \boldsymbol{\psi}_h)|$ is deemed sufficiently small.
The above methodology is compatible with the direct approach from Section 4.2.4. A typical workflow consists of computing a bijection $\mathbf{x}_h$ under the cost function (4.50) using the PDE-based approach and continuing to improve parametric quality using (4.53). Furthermore, once a bijective $\mathbf{x}_h$ has been found, it may serve as an initial guess for the direct approach from Section 4.2.4.

4.3.3. CHOICE OF ADJOINT BASIS

Problem (4.46) requires choosing a suitable dual spline space $\mathcal{V} \supset \bar{\mathcal{V}}_h \neq \mathcal{V}_h$, which typically results from uniformly refining $\mathcal{V}_h$ (in either h or the polynomial degree p), leading to a ~ 4-fold increase in the number of DOFs associated with the (linear) discrete adjoint equation. In a THB-setting, we have the luxury of choosing $\bar{\mathcal{V}}_h$ reminiscent of the role of K-refinement [HCB05] in a structured spline setting. Let (p, α) be the degree and regularity of $\mathcal{V}_h$ (which we assume to be equal in both directions for convenience) and let $\mathcal{T}$ denote the corresponding decomposition of $\hat{\Omega}$ into elements. We define $\bar{\mathcal{V}}_K(\mathcal{V}_h)$ as the richest (dimensionality-wise) THB space of degree $p+1$ and regularity $\alpha+1$ that is compatible with the elements in $\mathcal{T}$. Typically, we have $|\bar{\mathcal{V}}_K(\mathcal{V}_h)| \simeq |\mathcal{V}_h|$, where $|\cdot|$ denotes the dimension of a vector space.
While taking $\bar{\mathcal{V}}_h = \mathcal{V}_{h/2}$ yields more accurate adjoint solutions $\mathbf{z}_h \in (\bar{\mathcal{V}}_h^{\circ})^2$, we have found the choice $\bar{\mathcal{V}}_h = \bar{\mathcal{V}}_K(\mathcal{V}_h)$, to be sufficient for refinement based on both (4.49) and (4.53). As such, solving the discrete adjoint equation becomes a cheap operation.

4.3.4. REFINEMENT STRATEGIES

The decomposition into basis function wise contributions $\mathbf{r}_i(\mathbf{x}_h)$ introduced in (4.47) is particularly useful in a THB-setting since elementwise refinement may not change the dimension of the underlying THB spline space. In the following, we present several strategies for using $\mathbf{r}(\mathbf{x}_h)$ to mark basis functions $w_i \in [\mathcal{V}_h]$ for refinement. We define the vectors $\mathbf{w}$ and $\tilde{\mathbf{r}}$ with

$$\mathbf{w}_i = \int_{\hat{\Omega}} w_i \mathrm{d}S \quad \text{and} \quad \tilde{\mathbf{r}}_i = \frac{\mathbf{r}_i}{\mathbf{w}_i}. \tag{4.54}$$

Furthermore, we let $\tilde{\mathbf{r}}_{\max} = \max_i |\tilde{\mathbf{r}}_i|$ and $\mathcal{I} = \{1, \dots, |\mathcal{V}_h|\}$. Inspired by [PO03], we define

$$\mathcal{I}^{\alpha}_{\max} = \{ i \in \mathcal{I} \mid |\tilde{\mathbf{r}}_i| \geq \beta \tilde{\mathbf{r}}_{\max} \} \tag{4.55}$$

as the index-set of absolutely weighted contributions that exceed the value $\beta\tilde{\mathbf{r}}_{\max}$, for some $\beta \in [0,1]$. The $i \in \mathcal{I}^{\alpha}_{\max}$ then constitute the indices corresponding to basis functions whose supporting elements $\mathcal{E}^k \in \mathcal{T}$, from the k-th level in the element hierarchy, are replaced by finer counterparts $\mathcal{E}^{k+1}$ from the $(k+1)$-th level. Note that the function may, due to preceding refinements of other functions, be already partially supported by $\mathcal{E}^l \in \mathcal{T}$, with $l \geq k+1$. In this case only the coarsest supporting elements $\mathcal{E}^k$ are refined. As a result, upon constructing the canonical THB-spline space over the refined $\mathcal{T}$, $w_i \in [\mathcal{V}_h]$ is replaced by several functions from the next level in the hierarchy, leading to a local increase of the DOFs. Basis function wise refinement ensures that always at least one function is removed from the basis and replaced by several finer ones.
Since both (4.50) and (4.53) are strictly positive quantities of interest, disregarding negative contributions in (4.54) is a plausible strategy, too. Heuristically, this strategy mildly reduces the total number of required DOFs until bijectivity is achieved. However, this comes at the expense of a larger number of the required a posteriori refinements, which are limited to typically no more than $3-4$ using (4.55).

4.3.5. RESULTS

To demonstrate the appeal of local refinement made possible by THB splines, in the following, we present parameterizations for the U.S. state of Indiana, the German province of North Rhine-Westphalia and the country of Austria, all of which have complicated boundaries but relatively simple interior. The initial basis $[\mathcal{V}_h]$ results from refining an initial grid comprised of 7×7 elements by the boundaries until the contours of Ω are approximated sufficiently well. In all cases, $\mathcal{V}_h$ is a bicubic hierarchical space. We take $\bar{\mathcal{V}}_h = \bar{\mathcal{V}}_K(\mathcal{V}_h)$ (see Section 4.3.3) and base refinement on (4.55) with $\beta = 0.2$.

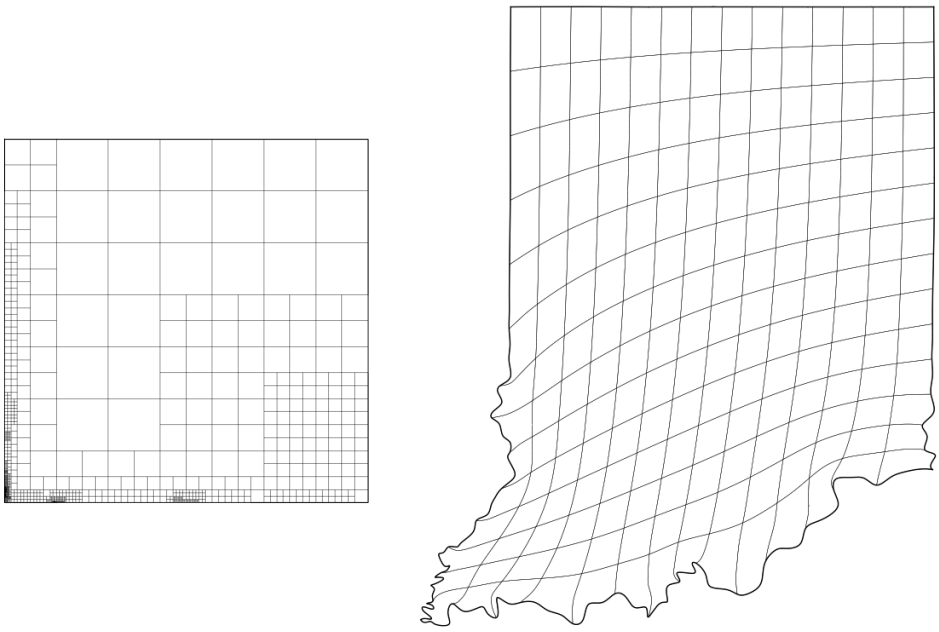


Figure 4.4: The domain with canonical bicubic basis of 2338 DOFs (left) and the THB-spline parameterization of the U.S. state of Indiana (right).

Figures (4.4) to (4.6) clearly demonstrate the DOF savings made possible by local refinement. Not surprisingly, refinement especially affects the protruded and concave areas close to the boundaries.

At every refinement level, parameterizations were computed using the Newton-Krylov approach from Section 4.2.1. They were post-processed with the direct approach from Section 4.2.4 once bijectivity had been achieved.

The iterative solver typically converges after $4-5$ nonlinear iterations on the coarsest level plus another $2-3$ iterations per a posteriori refinement. Once bijectivity is achieved, initializing the direct approach from (4.39) with the PDE solution typically leads to convergence after fewer than 3 iterations.

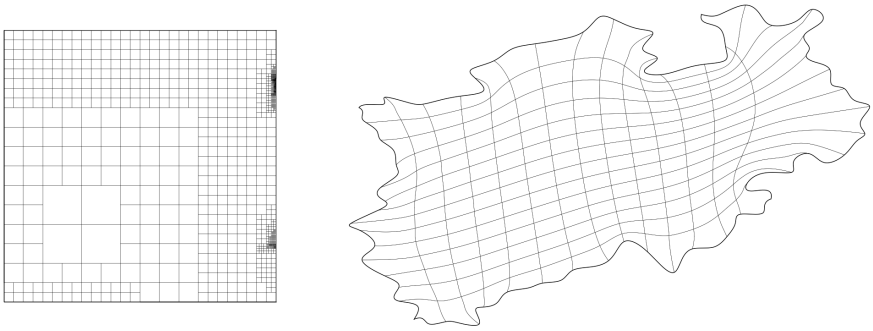


Figure 4.5: The domain with bicubic basis of 2676 DOFs (left) and the THB-spline parameterization of the German province of North Rhine-Westphalia (right).

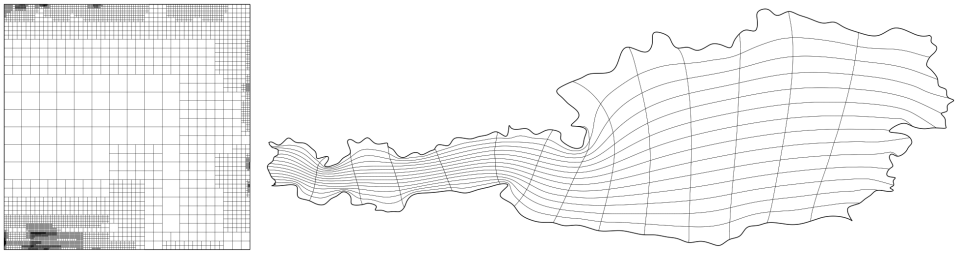


Figure 4.6: The domain with bicubic basis comprised of 9640 DOFs (left) and the THB-spline parameterization of Austria (right).

4.4. DOMAIN OPTIMIZATION

As demonstrated in Section 4.3.5, the approach from Section 4.3 can handle challenging geometries. However, it lacks the flexibility of precisely controlling the parametric properties of the outcome, which may lead to undesirable features, such as large elements (see Figures 4.2 and 4.3). As such, in the following we present a framework that allows for more flexibility, where we pay particular attention to mitigating the aforementioned pathologies associated with inversely harmonic maps.

Instead of mapping inversely harmonically into a domain $\hat{\Omega}$ with a Cartesian coordinate system, we now define it through a parameterization $\mathbf{s} : \hat{\Omega} \to \hat{\Omega}$. For convenience, we assume that the boundary correspondence $\mathbf{s}|_{\partial\hat{\Omega}} : \partial\hat{\Omega} \to \partial\hat{\Omega}$ is the identity. Suppose that $\mathbf{x}^* : \hat{\Omega} \to \Omega$ solves the equation

$$\Delta_{\mathbf{x}}\boldsymbol{\xi} = \mathbf{0}, \quad \text{s.t.} \quad \mathbf{x}|_{\partial\hat{\Omega}} = \mathbf{x}_D(\boldsymbol{\xi}), \tag{4.56}$$

for $\mathbf{x}$. Then, if $\mathbf{x}(\boldsymbol{\xi})$ is the solution of

$$\Delta_{\mathbf{x}}\mathbf{s}(\boldsymbol{\xi}) = \mathbf{0}, \quad \text{s.t.} \quad \mathbf{x}|_{\partial\hat{\Omega}} = \mathbf{x}_D(\mathbf{s}(\boldsymbol{\xi})), \tag{4.57}$$

it clearly satisfies $\mathbf{x} = \mathbf{x}^* \circ \mathbf{s}$, thanks to the fact that $\mathbf{x}_D \circ \mathbf{s} = \mathbf{x}_D$ on $\partial\hat{\Omega}$ (i.e., the boundary condition does not change upon pullback). As such, we may approximate compositions

$\mathbf{x}^* \circ \mathbf{s}$ by solving the discretized counterpart of (4.57).
Introducing the set of vectors

$$\mathbf{p}^{ij}(\mathbf{s}) = -T^{-1}\frac{\partial^2 \mathbf{s}}{\partial \boldsymbol{\xi}_i \partial \boldsymbol{\xi}_j}, \quad \text{with} \quad T = \partial_{\boldsymbol{\xi}}\mathbf{s} \quad \text{and} \quad (i,j) \in \{1,2\}\times\{1,2\}, \tag{4.58}$$

it can be shown that with $\mathbf{s} = \mathbf{s}(\boldsymbol{\xi})$, (4.57) can be reformulated as [TSW98, Chapter 4]

$$A(\mathbf{x}): \left(H(\mathbf{x}_i) + P^1(\mathbf{s})\frac{\partial \mathbf{x}_i}{\partial \xi} + P^2(\mathbf{s})\frac{\partial \mathbf{x}_i}{\partial \eta}\right) = 0 \quad i \in \{1,2\}, \quad \text{s.t.} \quad \mathbf{x}|_{\partial\hat{\Omega}} = \mathbf{x}_D|_{\partial\hat{\Omega}}. \tag{4.59}$$

Here, the matrices P^1 and P^2 satisfy

$$P^k_{ij}(\mathbf{s}) = \mathbf{p}^{ij}_k(\mathbf{s}), \quad k \in \{1,2\}. \tag{4.60}$$

Therefore, we introduce

$$F(\mathbf{x},\boldsymbol{\sigma},\mathbf{s}) = \sum_{i=1}^{2}\int_{\hat{\Omega}} \boldsymbol{\tau}(\boldsymbol{\sigma},\mathbf{x})_i A(\mathbf{x}) : \underbrace{\left(H(\mathbf{x}_i) + P^1(\mathbf{s})\frac{\partial \mathbf{x}_i}{\partial \xi} + P^2(\mathbf{s})\frac{\partial \mathbf{x}_i}{\partial \eta}\right)\det T(\mathbf{s})}_{\tilde{H}(\mathbf{x}_i,\mathbf{s})}\,\mathrm{d}S, \tag{4.61}$$

and for given $\mathbf{s}(\boldsymbol{\xi})$, we solve

$$\text{find } \mathbf{x}_h \in \mathscr{U}_h^{\mathbf{x}_D} \quad \text{s.t.} \quad F(\mathbf{x}_h,\boldsymbol{\sigma}_h,\mathbf{s}) = 0 \quad \forall \boldsymbol{\sigma}_h \in \mathscr{U}_h^{\mathbf{0}}, \tag{4.62}$$

in order to approximate $\mathbf{x}^* \circ \mathbf{s}$. Unless stated otherwise, we utilize the Newton approach from Section 4.2.1 with $\boldsymbol{\tau}(\boldsymbol{\sigma},\mathbf{x}) = \boldsymbol{\sigma}$. We can apply the Picard approach from Section 4.2.3 by replacing $H(\mathbf{x}_i) \to \tilde{H}(\mathbf{x}_i,\mathbf{s})$ in equation (4.34). In the following, we present several strategies for choosing $\mathbf{s}$ to improve the parametric properties of the composite mapping.

4.4.1. EXPLOITING THE MAXIMUM PRINCIPLE

Clearly, for well-posedness of (4.62), $\mathbf{s}: \hat{\Omega} \to \hat{\Omega}$ should not fold. As the control mapping maps into a convex domain, we may exploit the fact that if it is the solution to a second order elliptic problem in divergence form, it is necessarily a bijection [BMN01]. Thus, let $\mathbf{s} = (\mathbf{s}_1,\mathbf{s}_2)^T$ be such that

$$\nabla_{\boldsymbol{\xi}} \cdot \left(D\nabla_{\boldsymbol{\xi}}\mathbf{s}_i\right) = 0 \quad i \in \{1,2\}, \quad \text{in} \quad \hat{\Omega}, \quad \text{s.t.} \quad \mathbf{s}(\boldsymbol{\xi}) = \boldsymbol{\xi} \text{ on } \partial\hat{\Omega}, \tag{4.63}$$

where $D: \hat{\Omega} \to \mathbb{R}^{2\times 2}$ is an SPD diffusivity tensor. In the following, we assume that an accurate approximation $\mathbf{x}_h^*$ of $\mathbf{x}^*$ has been computed using the methodology from Section 4.3. In order to mitigate the impact of the well-known pathologies of inversely harmonic maps (see Section 4.2), we may select D in (4.63) such that the value of

$$L_{\text{Area}}(\mathbf{x}_h) = \int_{\hat{\Omega}} \det J(\mathbf{x}_h)^2 \mathrm{d}S \tag{4.64}$$

is expected to decrease with respect to $\mathbf{x}^*$ (see (4.10)). Note that

$$\begin{aligned}\left(\det J(\mathbf{x}^* \circ \mathbf{s})\right)^2 &\simeq \left(\det \partial_{\mathbf{s}} \mathbf{x}_h^*\right)^2 \det J(\mathbf{s})^2 \\ &= \left(\det \partial_{\mathbf{s}} \mathbf{x}_h^*\right)^2 \left(g_{11} g_{22} - g_{12}^2\right)_{\boldsymbol{\xi} \to \mathbf{s}} \\ &\leq \frac{1}{2} \left(\det \partial_{\mathbf{s}} \mathbf{x}_h^*\right)^2 \left(g_{11} + g_{22}\right)_{\boldsymbol{\xi} \to \mathbf{s}}^2, \end{aligned} \tag{4.65}$$

where the subscript $\boldsymbol{\xi} \to \mathbf{s}$ indicates that the g_{ij} between brackets refer to the metric induced by $\mathbf{s}(\boldsymbol{\xi})$. Given that $\mathbf{s}(\boldsymbol{\xi}) = \boldsymbol{\xi}$ initially, (4.65) suggests a convex optimization problem of the form

$$L_{\text{PoissonArea}}(\mathbf{s}, k) \to \min_{\mathbf{s} \in \mathcal{V}_h^2}, \quad \text{s.t.} \quad \mathbf{s}(\boldsymbol{\xi}) = \boldsymbol{\xi} \text{ on } \partial\hat{\Omega}, \tag{4.66}$$

where

$$L_{\text{PoissonArea}}(\mathbf{s}, k) = \int_{\hat{\Omega}} \left(\det \partial_{\boldsymbol{\xi}} \mathbf{x}_h^*\right)^k \left(\|\partial_{\boldsymbol{\xi}} \mathbf{s}_1\|^2 + \|\partial_{\boldsymbol{\xi}} \mathbf{s}_2\|^2\right) \mathrm{d}S, \tag{4.67}$$

for recomputing $\mathbf{s}(\boldsymbol{\xi})$. As such, we are solving the discretized equations corresponding to (4.63) with

$$D = \left(\det \partial_{\boldsymbol{\xi}} \mathbf{x}_h^*\right)^k I^{2\times 2}. \tag{4.68}$$

Even though the exact solution of (4.63) does not fold, the discretized counterpart may fold due to extreme diffusive anisotropy. This can be counteracted by reducing the value of k. Alternatively, (4.67) can be utilized for DWR-based a posteriori refinement to achieve bijectivity and accuracy of $\mathbf{s} : \hat{\Omega} \to \hat{\Omega}$.
Upon completion, we compute $\mathbf{x}_h \in \mathcal{V}_h^2$ using the control mapping $\mathbf{s} : \hat{\Omega} \to \hat{\Omega}$, with a posteriori refinement if necessary.

k	0	0.5	1	1.5
$L_{\text{Area}}(\mathbf{x}_h) \times 10^{-2}$	3.291	2.077	1.439	1.299

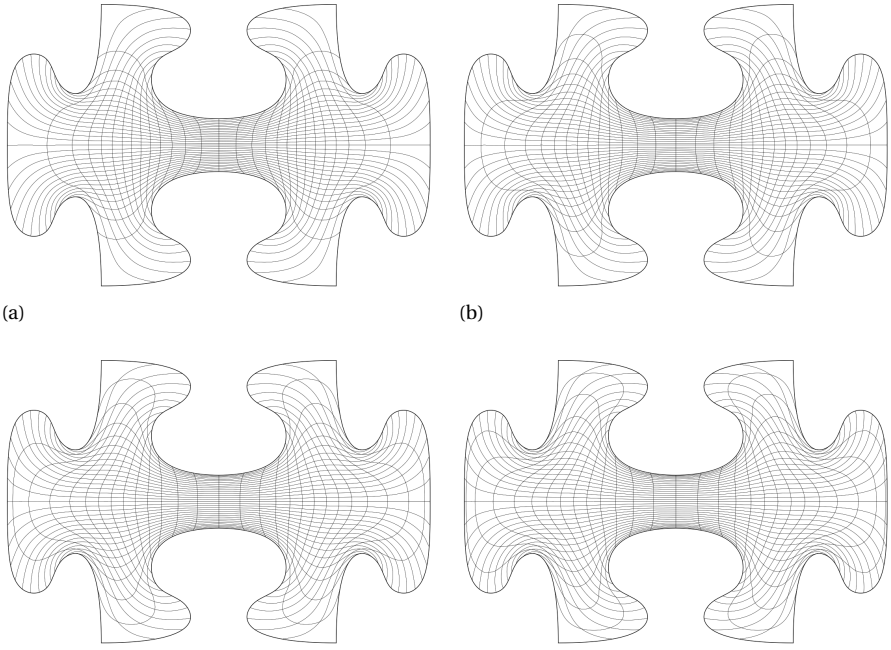
Table 4.2: Evaluation of $L_{\text{Area}}(\mathbf{x}_h)$ for various values of k.

Figure 4.7 shows puzzle piece geometry parameterizations for various values of k, while Table 4.2 contains the outcomes of substituting into (4.64). Both clearly demonstrate that the methodology has the desired effect, with more drastic outcomes for larger values of k. Figure 4.8 shows the isolines of $\mathbf{s}(\boldsymbol{\xi})$ before and after reparameterization with $k = 1.5$. All parameterizations were computed with the reference basis corresponding to Figure 4.7a. No a posteriori refinements were necessary.
Figure 4.9 shows parameterizations of the U.S. state of Indiana for $k = 0$ and $k = 1$. Contrary to Table 4.2, with

$$L_{\text{Area}}(\mathbf{x}_h) = 1.049 \times 10^2 \quad \text{for} \quad k = 0 \quad \text{and} \quad L_{\text{Area}}(\mathbf{x}_h) = 1.008 \times 10^2 \quad \text{for} \quad k = 1,$$

the effect is very mild. Restricting the integrals to $\eta < 1/7$, however, the difference becomes more pronounced with

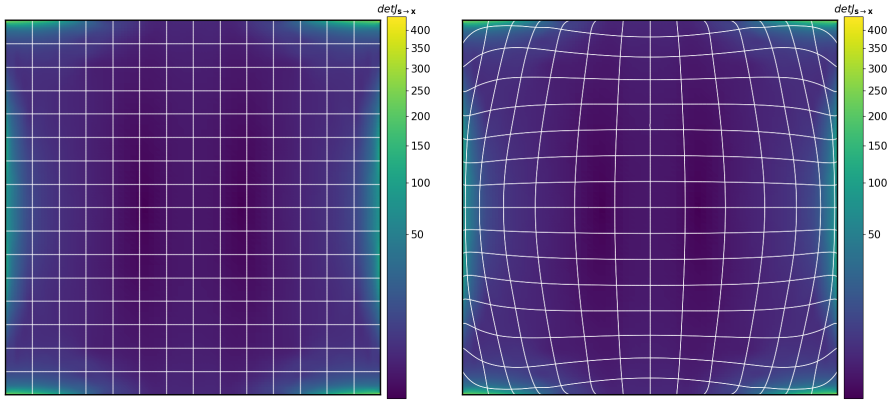
$$L_{\text{Area}}(\mathbf{x}_h) = 17.167 \quad \text{and} \quad L_{\text{Area}}(\mathbf{x}_h) = 14.118,$$



(a) (b)

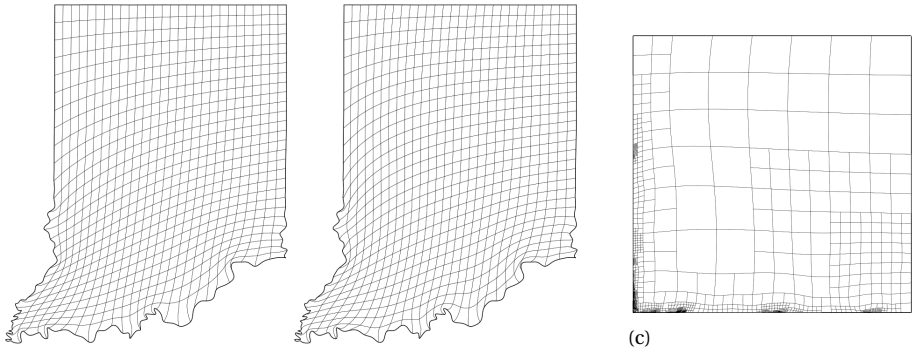
(c) (d)

Figure 4.7: Several parameterizations of the puzzle piece with reparameterization based on (4.63) and (4.68) with the reference parameterization $k = 0$ (a), reparameterization with $k = 0.5$ (b), $k = 1$ (c) and $k = 1.5$ (d).



(a) (b)

Figure 4.8: Plots showing the reference domain (a) and the reparameterized domain based on (4.63) and (4.68) with $k = 1.5$ (b). The figure clearly shows that the elements are contracted wherever $\det \partial_{\boldsymbol{\xi}} \mathbf{x}_h^*$ is large.



(a) (b) (c)

Figure 4.9: Parameterization of the U.S. state of Indiana with $k = 0$ (a), $k = 1$ (b) and the corresponding reparameterized domain (c).

respectively. Unsurprisingly from the shape of the geometry, the difference is most striking close to the lower boundary, which can also be seen in the figure. A posteriori refinement was necessary in Figure 4.9b.
Heuristically, reparameterization based on the maximum principle proves to be one of the most robust and effective choices for a wide range of geometries while being computationally efficient. This is thanks to the fact that it addresses the known pathologies of inversely harmonic maps, while also yielding smooth solutions, which preserves smoothness of the composite mapping.

4.4.2. CONSTRAINED DOMAIN OPTIMIZATION

The concept of reparameterizing the domain in order to alter the parametric properties of the recomputed geometry parameterization can be further extended in a way more reminiscent of the well-known cost function minimization approach (see Section 4.1). Given an accurate approximation $\mathbf{x}_h^*$ of $\mathbf{x}^*$ (see Section 4.4.1), we define the metric $G_{\mathbf{s}\to\mathbf{x}} = \partial_{\mathbf{s}}\mathbf{x}^T \partial_{\mathbf{s}}\mathbf{x}$, which is initially given by

$$G_{\mathbf{s}\to\mathbf{x}} = \partial_{\boldsymbol{\xi}}\mathbf{x}^{*T} \partial_{\boldsymbol{\xi}}\mathbf{x}^* \simeq \partial_{\boldsymbol{\xi}}\mathbf{x}_h^{*T} \partial_{\boldsymbol{\xi}}\mathbf{x}_h^*.$$

Hence, in order to optimize $\mathbf{x}_h(\boldsymbol{\xi})$, we optimize $\mathbf{s}(\boldsymbol{\xi})$ in the metric induced by $G_{\mathbf{s}\to\mathbf{x}}$. With

$$g_{ij}^{\mathbf{s}} = \left[\partial_{\boldsymbol{\xi}}\mathbf{s}^T G_{\mathbf{s}\to\mathbf{x}} \partial_{\boldsymbol{\xi}}\mathbf{s}\right]_{ij} \quad \text{and} \quad J_{ij}^{\mathbf{s}} = (\partial_{\boldsymbol{\xi}}\mathbf{x}_h^* \partial_{\boldsymbol{\xi}}\mathbf{s})_{ij}, \tag{4.69}$$

we define domain optimization cost functions $Q_i^{\mathbf{s}}(\mathbf{s})$ by replacing $g_{ij} \to g_{ij}^{\mathbf{s}}$ and $J_{ij} \to J_{ij}^{\mathbf{s}}$ in the Q_i introduced in equation (4.4) (see Section 4.1). We may nevertheless choose to add terms of the form $Q_i(\mathbf{s})$, which should then be regarded as regularization terms. Let $\mathscr{U}_h^{\square} = \{\mathbf{v} \in \mathcal{V}_h^2 \mid \mathbf{v} = \boldsymbol{\xi} \text{ on } \partial\hat{\Omega}\}$. A domain optimization problem takes the form

$$\int_{\hat{\Omega}} Q(\mathbf{s}) \mathrm{d}S \to \min_{\mathbf{s}\in\mathscr{U}_h^{\square}}, \quad \text{s.t.} \quad \mathbf{C}(\mathbf{s}) \geq \mathbf{0}, \tag{4.70}$$

with

$$Q(\mathbf{s}) = \sum_i \lambda_i^{\mathbf{s}} Q_i^{\mathbf{s}}(\mathbf{s}) + \sum_j \lambda_j Q_j(\mathbf{s}). \tag{4.71}$$

Here, the constraint $\mathbf{C}(\mathbf{s}) \geq 0$ ensures that the minimizer of (4.70) does not fold. In the following, we list all choices of $\mathbf{C}(\mathbf{s})$ that come to mind.
Given the element segmentation $\mathscr{T}$ of $\hat{\Omega}$, by $\mathcal{V}^{p,\alpha}(\mathscr{T})$ we denote the canonical THB-space with order p and regularity α that is compatible with $\mathscr{T}$. Note that $\alpha \leq p-1$. Clearly, if $\mathcal{V}_h$ has order p and regularity $\alpha \leq p-1$, this implies that

$$\det \partial_{\boldsymbol{\xi}} \mathbf{s} \in \mathcal{V}^{2p-1,\alpha-2}(\mathscr{T}).$$

As such, we also have

$$\det \partial_{\boldsymbol{\xi}} \mathbf{s} \in \mathcal{V}^{2p-1,-1}(\mathscr{T}).$$

Hence, we can base the constraint on Bézier extraction, in which we require that all weights of projecting $\det \partial_{\boldsymbol{\xi}} \mathbf{s}$ onto $\mathcal{V}^{2p-1,-1}(\mathscr{T})$ be positive. Let $\hat{\mathbf{d}}$ be the corresponding vector of weights. We have

$$\hat{\mathbf{d}}(\mathbf{s}) = \hat{M}^{-1}\hat{\mathbf{f}}(\mathbf{s}) > \mathbf{0}, \quad \text{where} \quad \hat{\mathbf{f}}_i(\mathbf{s}) = \int_{\hat{\Omega}} \hat{\phi}_i \det \partial_{\boldsymbol{\xi}} \mathbf{s} \mathrm{d}S, \tag{4.72}$$

with

$$\hat{\phi}_i \in \left[\mathcal{V}^{2p-1,-1}(\mathscr{T})\right] \quad \text{and} \quad \hat{M}_{i,j} = \int_{\hat{\Omega}} \hat{\phi}_i \hat{\phi}_j \mathrm{d}S. \tag{4.73}$$

Note that $\hat{M}$ is block-diagonal with $|\mathscr{T}|$ blocks of size $(2p, 2p)$. Hence, we computationally efficiently assemble $\hat{M}^{-1}$ simply by computing the inverse of all separate blocks leading to a sparse block-diagonal matrix. As such, the computational costs of testing whether the condition $\hat{\mathbf{d}} > \mathbf{0}$ is fulfilled reduces to the assembly of $\hat{\mathbf{f}}$ along with one sparse matrix-vector multiplication. Assembly of the constraint gradient of $\hat{\mathbf{d}}(\mathbf{c}_{\mathscr{I}})$, where $\mathbf{c}_{\mathscr{I}}$ is a vector containing the inner control points of $\mathbf{s}$, requires the assembly of $\partial_{\mathbf{c}_{\mathscr{I}}} \hat{\mathbf{f}}$ and a sparse matrix-matrix multiplication. The assembly is hence feasible. However, for large values of p this may lead to an infeasibly large number of constraints.
Inspired by [GENN12], we formulate an alternative constraint by projecting $\det \partial_{\boldsymbol{\xi}} \mathbf{s}$ onto the coarser THB-space $\mathcal{V}^{2p,\alpha-2}(\mathscr{T})$. Similar to (4.72), this leads to a constraint of the form

$$\mathbf{d}(\mathbf{s}) = M^{-1}\mathbf{f}(\mathbf{s}) > \mathbf{0}, \quad \text{where} \quad \mathbf{f}_i(\mathbf{s}) = \int_{\hat{\Omega}} \phi_i \det \partial_{\boldsymbol{\xi}} \mathbf{s} \mathrm{d}S, \tag{4.74}$$

with

$$\phi_i \in \left[\mathcal{V}^{2p-1,\alpha-2}(\mathscr{T})\right] \quad \text{and} \quad M_{i,j} = \int_{\hat{\Omega}} \phi_i \phi_j \mathrm{d}S. \tag{4.75}$$

Increasing the values of p and α, unlike for (4.72), the length of $\mathbf{d}$ in (4.74) increases only slowly (thanks to K-refinement). On the other hand, the matrix M is not block-diagonal and neither is it separable (unlike in a structured spline setting). As such, the assembly of the constraint gradient is prohibitively expensive. Here, a remedy is to introduce the vector of slack variables $\mathbf{e} > 0$. The constraint from (4.74) can be reformulated as follows:

$$C_\alpha(\mathbf{s},\mathbf{e}) = \mathbf{f}(\mathbf{s}) - M\mathbf{e} = \mathbf{0}, \quad \text{with} \quad \mathbf{e} > \mathbf{0}. \tag{4.76}$$

Hence, we avoid inversion with M at the expense of introducing an additional inequality constraint and changing the existing inequality constraint to an equality constraint. Note that we have:

$$\frac{\partial C_\alpha(\mathbf{s},\mathbf{e})}{\partial(\mathbf{c}_\mathscr{J},\mathbf{e})} = \left[\frac{\partial \mathbf{f}}{\partial \mathbf{c}_\mathscr{J}}, -M\right] \quad \text{and} \quad \frac{\partial \mathbf{e}}{\partial(\mathbf{c}_\mathscr{J},\mathbf{e})} = [0, I], \tag{4.77}$$

where I denotes the identity matrix of appropriate dimension.
Given a set of abscissae $\Xi = \{\boldsymbol{\xi}_1^c, \dots \boldsymbol{\xi}_m^c\} \subset \mathbb{R}^2$, an alternative constraint $\mathbf{C}^\Xi(\mathbf{s})$ follows from requiring that

$$\boldsymbol{\epsilon}_i^L \le \det \partial_{\boldsymbol{\xi}} \mathbf{s}(\boldsymbol{\xi}_i^c) \le \boldsymbol{\epsilon}_i^U, \quad \forall i \in \{1,\dots,m\}, \tag{4.78}$$

where $\mathbb{R}^m \ni \boldsymbol{\epsilon}^{L,U} \ge 0$ are lower and upper thresholds. Note that (4.78) is nonlinear and nonconvex but not a sufficient condition for bijectivity of $\mathbf{s}$. However, it makes bijectivity likely for m sufficiently large.
Finally, assuming that $\mathbf{s}$ is built from a structured basis $[\mathcal{V}_h]$ resulting from a tensor product of the univariate bases

$$\{N_1^\square, \dots, N_n^\square\} \quad \text{and} \quad \{M_1^\square, \dots, M_m^\square\},$$

we may alternatively utilize the linear constraint proposed in [XMDG11]. Typically, we take $[\mathcal{V}_h]$ as the cardinality-wise largest structured basis compatible with $\mathscr{T}$. Given

$$\mathbf{s}(\boldsymbol{\xi}) = \sum_{i,j} \mathbf{c}_{i,j} N_i^\square(\xi) M_j^\square(\eta), \tag{4.79}$$

let the cones $\mathscr{C}_1(\mathbf{s})$ and $\mathscr{C}_2(\mathbf{s})$ be generated by the half rays $\mathbb{R}^+\Delta_{i,j}^1$ and $\mathbb{R}^+\Delta_{i,j}^2$ with

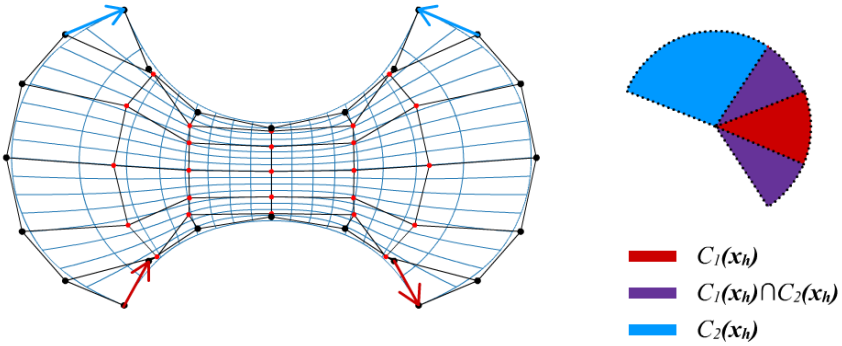
$$\Delta_{i,j}^1 = \mathbf{c}_{i+1,j} - \mathbf{c}_{i,j} \quad \text{and} \quad \Delta_{i,j}^2 = \mathbf{c}_{i,j+1} - \mathbf{c}_{i,j},$$

respectively. The constraint is based on the observation that if $\mathscr{C}_1(\mathbf{s})$ and $\mathscr{C}_2(\mathbf{s})$ only intersect in $\boldsymbol{\xi} = \mathbf{0}$, then $\mathbf{s}$ is bijective. In a direct optimization of $\mathbf{x}_h$, above constraint may be impractical since for most $\mathbf{x}_D$, the set

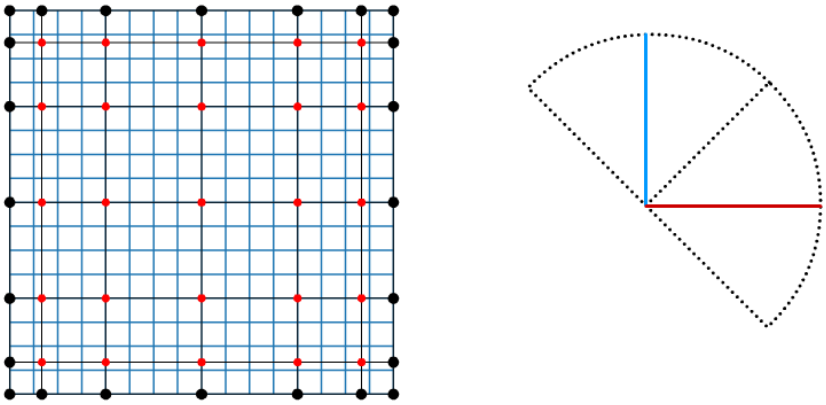
$$\left\{\mathbf{x}_h \in \mathscr{U}_h^{\mathbf{x}_D} \mid \mathscr{C}_1(\mathbf{x}_h) \cap \mathscr{C}_2(\mathbf{x}_h) = \{\mathbf{0}\}\right\}$$

is empty or the constraint is too restrictive. However, in the case of optimizing $\mathbf{s}$, for $\mathbf{s}^0 = \boldsymbol{\xi}$, the cones $\mathscr{C}_1(\mathbf{s}^0)$ and $\mathscr{C}_2(\mathbf{s}^0)$ are generated by $R^+(1,0)^T$ and $R^+(0,1)^T$, respectively. A linear constraint $\mathbf{C}^L(\mathbf{s})$ follows from requiring that $\mathscr{C}_1(\mathbf{s})$ and $\mathscr{C}_2(\mathbf{s})$ be contained in the cones generated by

$$\left\{\mathbb{R}^+ \times (1,-1+\epsilon)^T, \mathbb{R}^+ \times (1,1-\epsilon)^T\right\} \quad \text{and} \quad \left\{\mathbb{R}^+ \times (1,1+\epsilon)^T, \mathbb{R}^+ \times (-1,1-\epsilon)^T\right\},$$



(a) In this geometry, the constraint is violated despite the bijectivity of the mapping, demonstrating the restrictiveness of $\mathbf{C}^L(\mathbf{x}_h)$.



(b) In the case of the unreparameterized domain, the linear constraint is much less restrictive.

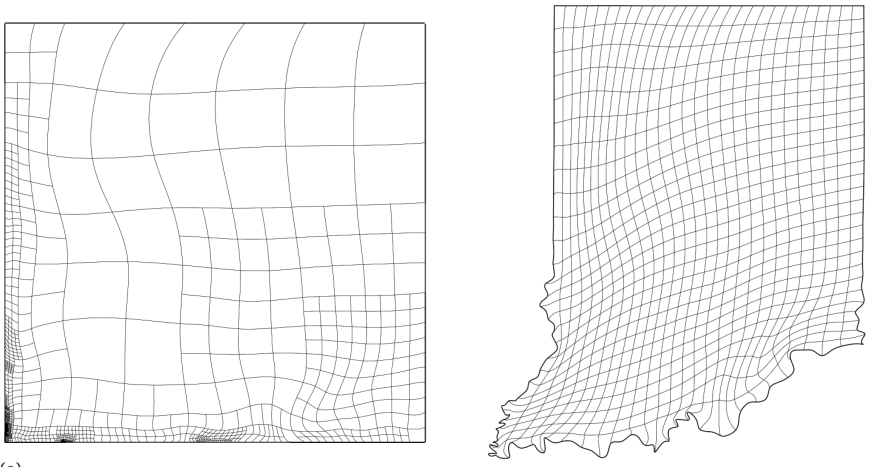
Figure 4.10: Depiction of the bijectivity constraint $\mathbf{C}^L$. In (a), the constraint is violated since $\mathscr{C}^1(\mathbf{x}_h) \cap \mathscr{C}^2(\mathbf{x}_h) \neq \{\mathbf{0}\}$ despite the bijectivity of the mapping. In (b) we see that the Cartesian domain is located exactly in the center of the feasible region generated by the cones with $\theta(\Delta^1_{i,j}) \in (-\pi/4, \pi/4)$ and $\theta(\Delta^2_{i,j}) \in (\pi/4, 3\pi/4)$.

respectively. Here $\epsilon \ll 1$ is a small positive parameter. Clearly, $\mathbf{s}^0$ is located exactly in the center of the feasible region (see Figure 4.10), making the constraint much less restrictive at the expense of having to compute $\mathbf{x}^*_h$ first.

Remark. We can combine the proposed constraints with the principles from Section 4.4.1 to suppress overshoots due to extreme diffusive anisotropy. If $\mathbf{C}(\mathbf{s}) = \mathbf{C}_L(\mathbf{s})$, the problem remains convex.

Figure 4.11a shows the domain corresponding to the U.S. state of Indiana (see Figure 4.4) after optimizing with

$$Q = Q^{\mathbf{s}}_{\text{AreaOrthogonality}}$$



(a) (b)

Figure 4.11: Result of reparameterizing the domain corresponding to the U.S. state of Indiana with $Q = Q^{\mathbf{s}}_{\text{AreaOrthogonality}}$ (a) and the resulting recomputed geometry parameterization (b).

under the constraint $\mathbf{C}(\mathbf{s}) = \hat{\mathbf{d}}(\mathbf{s})$ (see equation (4.72)). The domain mapping $\mathbf{s}(\boldsymbol{\xi})$ is built from the same THB-basis as $\mathbf{x}_h^*$, comprised of 2338 DOFs. Since Newton failed to converge, we recomputed $\mathbf{x}_h$ using the Picard approach, which converged after 21 iterations. The result is depicted in Figure 4.11. No a posteriori refinements were required. The reparameterization reduces the value of $L_{\text{AreaOrthogonality}}$ from the initial

$$L_{\text{AreaOrthogonality}}(\mathbf{x}_h^*) = 1.77 \times 10^2, \quad \text{to} \quad L_{\text{AreaOrthogonality}}(\mathbf{x}_h) = 1.36 \times 10^2.$$

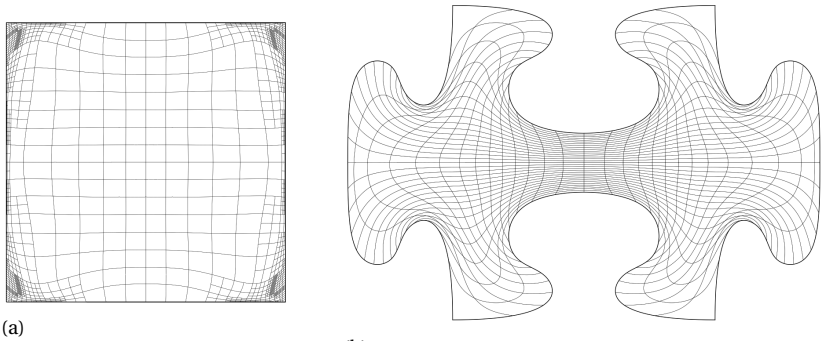
Next, we optimize the domain corresponding to the puzzle piece geometry (see Figure 4.2b) with $\mathbf{C}(\mathbf{s}) = \mathbf{C}_L(\mathbf{s})$ and $Q = Q^{\mathbf{s}}_{\text{Area}}$. Hereby, $\mathbf{s}(\boldsymbol{\xi})$ is built from a structured spline space comprised of 646 DOFs. The reparameterized domain is depicted in Figure 4.12a. Bijectivity of $\mathbf{x}_h$ is achieved with 2632 DOFs and the resulting parameterization is depicted in Figure 4.12b. With $L_{\text{Area}}(\mathbf{x}_h) = 142.710$, it is roughly as effective as the reparameterization from Figure 4.7 with $k = 1$.
Figure 4.13 shows the German province of North Rhine-Westphalia upon reparameterization with

$$Q = Q^{\mathbf{s}}_{\text{Orthogonality}},$$

where $\mathbf{s}(\boldsymbol{\xi})$ is built from a structured spline space comprised of 578 DOFs, with $\mathbf{C}(\mathbf{s}) = \mathbf{C}_L(\mathbf{s})$. Initially,

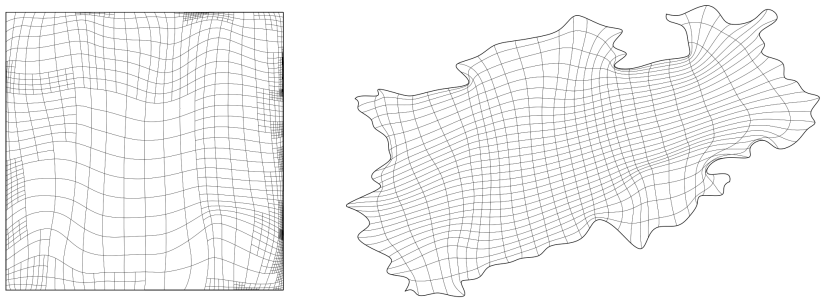
$$L_{\text{Orthogonality}}(\mathbf{x}_h^*) = 18.929, \quad \text{while} \quad L_{\text{Orthogonality}}(\mathbf{x}_h) = 5.160$$

upon recomputation. Bijectivity is achieved with 4584 DOFs, which is roughly double the initial 2724 DOFs.



(a) (b)

Figure 4.12: Result of optimizing the puzzle piece domain with $Q = Q^{\mathbf{s}}_{\text{Area}}$ under the constraint $\mathbf{C}(\mathbf{s}) = \mathbf{C}_L(\mathbf{s})$ (a) and the corresponding recomputed mapping (b).



(a) (b)

Figure 4.13: Result of reparameterizing the reference parameterization of the German province of North Rhine-Westphalia (see Figure 4.5), with $Q(\mathbf{s}) = Q^{\mathbf{s}}_{\text{Orthogonality}}(\mathbf{s})$. The reparameterized domain is shown in (a), while (b) shows the recomputed parameterization.

Finally, Figure 4.14 shows the result of reparameterizing the same geometry with

$$Q = Q^{\mathbf{s}}_{\text{AreaOrthogonality}}$$

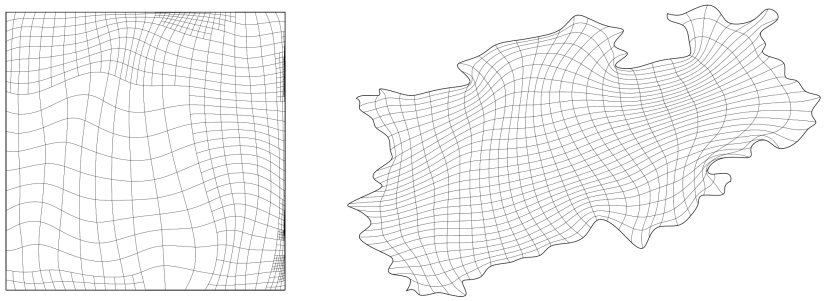
and the same constraints. Initially,

$$L_{\text{AreaOrthogonality}}(\mathbf{x}^*_h) = 51.244, \quad \text{while} \quad L_{\text{AreaOrthogonality}}(\mathbf{x}_h) = 30.896$$

upon recomputation. Bijectivity is achieved with only 2928 DOFs.

4.4.3. DIRECT OPTIMIZATION

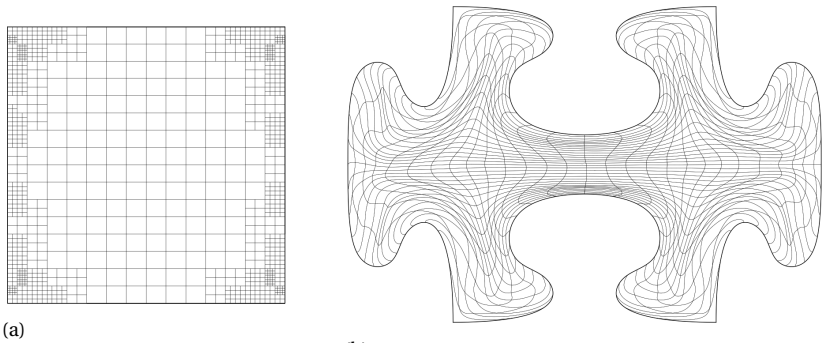
As an alternative to operating in the parametric domain, we may choose to directly optimize the geometry parameterization with respect to a quality cost function. As an advantage, we avoid the (possibly expensive) recomputation of $\mathbf{x}_h$. In order to avoid folding,



(a) (b)

Figure 4.14: Result of reparameterizing the reference parameterization of the German province of North Rhine-Westphalia (see Figure 4.5), with $Q(\mathbf{s}) = Q^{\mathbf{s}}_{\text{AreaOrthogonality}}(\mathbf{s})$. The reparameterized domain is shown in (a), while (b) shows the recomputed parameterization.

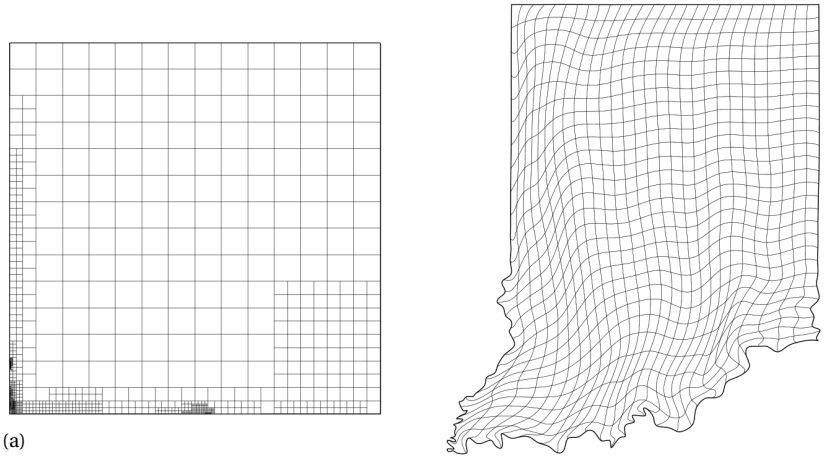
constraints should be employed. As a disadvantage, the linear constraint $\mathbf{C}_L(\mathbf{x}_h)$ cannot be used and the initial guess $\mathbf{x}_h^*$ may fail to satisfy the conditions $\hat{\mathbf{d}}(\mathbf{x}_h) > \mathbf{0}$ (cf. (4.72)) and $\mathbf{d}(\mathbf{x}_h) > \mathbf{0}$ (cf. (4.74)) despite being bijective. Heuristically, for complicated geometries, this is usually the case. In such cases, the only viable constraint is $\mathbf{C}^{\Xi}(\mathbf{x}_h)$ (cf. (4.78)).
We optimize the puzzle piece geometry with $Q(\mathbf{x}_h) = Q_{\text{Area}}(\mathbf{x}_h)$ under the constraint $\hat{\mathbf{d}}(\mathbf{x}_h) > 0$, where the initial guess $\mathbf{x}_h^*$ is the parameterization from Figure 4.7a. Figure 4.15



(a) (b)

Figure 4.15: The puzzle piece geometry after 21 iterations of minimizing $Q(\mathbf{x}_h) = Q_{\text{Area}}$ under the constraint $\hat{\mathbf{d}}(\mathbf{x}_h) > 0$ (b) and the corresponding domain (a). The minimization was initialized with the parameterization from Figure 4.7a.

shows the resulting parameterization. Convergence is achieved after 21 constrained iterations. The reparameterization reduces L_{Area} from the initial $L_{\text{Area}}(\mathbf{x}_h^*) = 3.29 \times 10^2$ to $L_{\text{Area}}(\mathbf{x}_h) = 0.96 \times 10^2$, which is slightly more pronounced than the reduction from Figure 4.7 with $k = 1.5$. However, the resulting parameterization is less regular compared to Figure 4.7d, which can be remedied by adding a regularization of the form $Q(\mathbf{x}_h) = Q_{\text{Area}}(\mathbf{x}_h) + \beta Q_{\text{Uniformity}}(\mathbf{x}_h)$. Next, we optimize the U.S. state of Indiana with



(a) (b)

Figure 4.16: The parameterization of the U.S. state of Indiana after 30 iterations of minimizing $Q(\mathbf{x}_h) = Q_{\text{Area}}$ (b) and the corresponding domain (a). The minimization was initialized with the parameterization from Figure 4.9a.

$Q(\mathbf{x}_h) = Q_{\text{Area}}(\mathbf{x}_h)$ under the constraint $\mathbf{C}^{\Xi}(\mathbf{x}_h) \geq 0$ with

$$\boldsymbol{\epsilon}_i^L = \alpha_L \times \det J(\mathbf{x}_h^*)(\boldsymbol{\xi}_i^c) \quad \text{and} \quad \boldsymbol{\epsilon}_i^U = \alpha_U \times \det J(\mathbf{x}_h^*)(\boldsymbol{\xi}_i^c), \tag{4.80}$$

(see equation (4.78)).
Figure 4.16 shows the resulting parameterization after 30 iterations. With

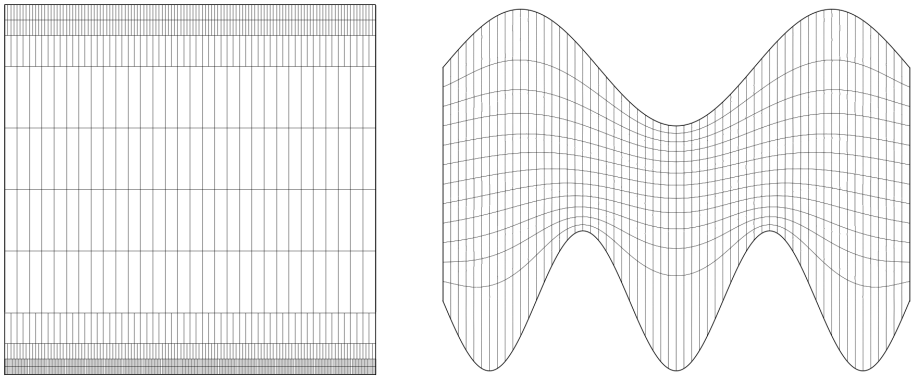
$$L_{\text{Area}}(\mathbf{x}_h^*) = 1.049 \times 10^2 \quad \text{and} \quad L_{\text{Area}}(\mathbf{x}_h) = 0.989 \times 10^2,$$

the reduction is mild, yet somewhat more pronounced than in Figure 4.9. Here, Ξ results from uniform sampling with 36 points per element. The choice of the relaxation factors $0 \leq \alpha_L \leq 1$ and $1 \leq \alpha_U$ in (4.80) tunes to which degree trading an increase in L_{Area} for a decrease in the employed cost function is acceptable. Here, more conservative choices lead to less cost function reduction but to more uniform cell sizes and vice versa. Furthermore, values of α_L closer to 1 prevent the grid from folding, even if fewer sampling points are used. We used $\alpha_L = 0.05$ and $\alpha_U = 4$.

4.4.4. ACHIEVING BOUNDARY ORTHOGONALITY

Many applications favor parameterizations with isolines that are orthogonal to the boundary contours. One way to achieve this is allowing $\lambda_i^{\mathbf{s}} = \lambda_i(\boldsymbol{\xi})^{\mathbf{s}}$ in (4.71) and taking $\lambda_{\text{Orthogonality}}^{\mathbf{s}}$ large close to $\partial\hat{\Omega}$. We are considering the example of achieving orthogonality at the northern and southern boundaries of the geometry depicted in Figure 4.17. To this end, we minimize the cost function

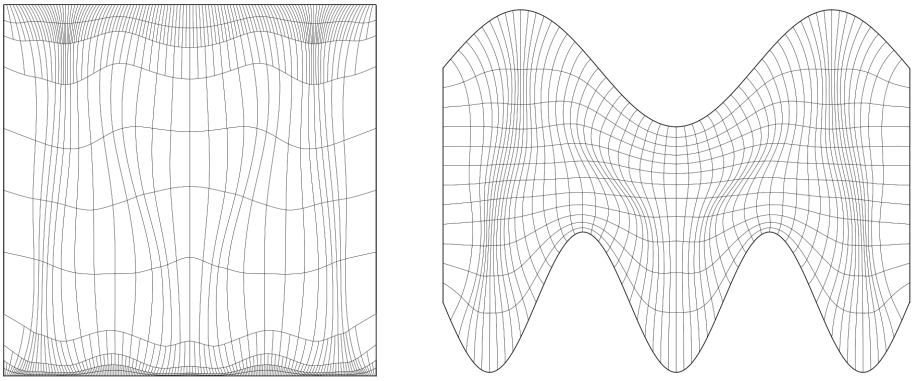
$$Q(\mathbf{s}) = (1 + \lambda_{\text{O}}(\boldsymbol{\xi}))Q_{\text{Orthogonality}}^{\mathbf{s}},$$



(a) (b)

Figure 4.17: Reference parameterization of a tube-like shaped geometry which is to be orthogonalized by the northern and southern boundaries.

where $\lambda_O(\boldsymbol{\xi})$ takes on large values close to the northern and southern boundaries of $\partial\hat{\Omega}$. We employ the constraint $\mathbf{C}(\mathbf{s}) = \mathbf{C}_L(\mathbf{s})$, where $\mathbf{s}(\boldsymbol{\xi})$ is built from a structured spline space comprised of 594 DOFs. The resulting paramterization is depicted in Figure 4.18. The



(a) (b)

Figure 4.18: Result of reprameterizing the geometry mapping from Figure 4.17 by weakly enforcing boundary orthogonality through a large penalty term (b) and the corresponding reparameterized domain (a).

figure indeed shows a large degree of orthogonalization, which is somewhat weaker in the protruded parts of the geometry. This is due to orthogonality only being enforced *weakly* through a penalty term. More pronounced boundary orthogonalization may be achieved by taking λ_O larger close to $\partial\hat{\Omega}$.
Let $\gamma_e, \gamma_w, \gamma_s$ and γ_n refer to the eastern, western, southern and northern parts of $\partial\hat{\Omega}$, respectively. For a more drastic boundary orthogonalization, we follow the approach

from [TSW98, Chapter 6], which consists of solving the problem

$$\Delta_{\mathbf{x}_h^*} f = 0 \quad \text{s.t.} \quad f = 0 \text{ on } \gamma_e, \;\; f = 1 \text{ on } \gamma_w \quad \text{and} \quad \frac{\partial f}{\partial \mathbf{n}} = 0 \text{ on } \gamma_s \cup \gamma_n \tag{4.81}$$

on an initially folding-free geometry parameterization $\mathbf{x}_h^*$. Here, $\mathbf{n}$ denotes the unit outward normal vector on $\partial\Omega$. Upon completion, the control mapping $\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2)^T \equiv (s,t)^T$ is computed from

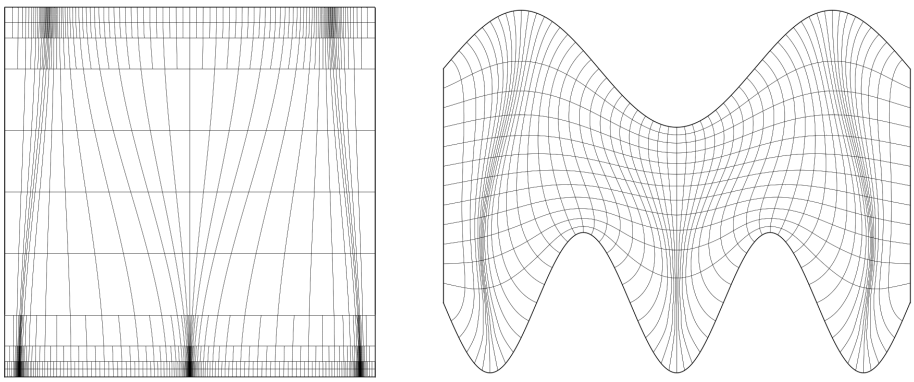
$$s(\xi,\eta) = f(\xi,0)\,H_0(\eta) + f(\xi,1)\,H_1(\eta) \quad \text{and} \quad t(\xi,\eta) = \eta, \tag{4.82}$$

where

$$H_0(\eta) = (1+2\eta)(1-\eta)^2 \quad \text{and} \quad H_1(\eta) = (3-2\eta)\eta^2 \tag{4.83}$$

are cubic Hermite interpolation functions. It can be shown that with this choice of s and t, the solution of (4.57) is orthogonal at γ_s and γ_n. We approximately solve for f by computing the solution f_h of the discretized counterpart of (4.81) over some structured spline space $\mathcal{V}_h$. Hereby, the Neumann boundary conditions are weakly imposed through partial integration. The control mapping follows from replacing $f \to f_h$ in (4.82). Should orthogonality at γ_w and γ_e be desired, we simply exchange the roles of $s \to t$, $(\gamma_s, \gamma_n) \to (\gamma_w, \gamma_e)$ and $\xi \to \eta$.

Remark. Unlike f, f_h may fail to be monotone increasing on γ_s or γ_n, leading to a folded control mapping $\mathbf{s}(\boldsymbol{\xi})$.



(a) (b)

Figure 4.19: Result of reprameterizing the geometry mapping from Figure 4.17 using the approach proposed in [TSW98, Chapter 6] (b) and the corresponding reparameterized domain (a).

Figure 4.19 shows the recomputed parameterization of the same geometry using the preceding methodology, along with the reparameterized parametric domain, which has been computed from the same structured spline basis as in Figure (4.18). The figure shows an outstanding boundary orthogonalization, which comes at the expense of larger

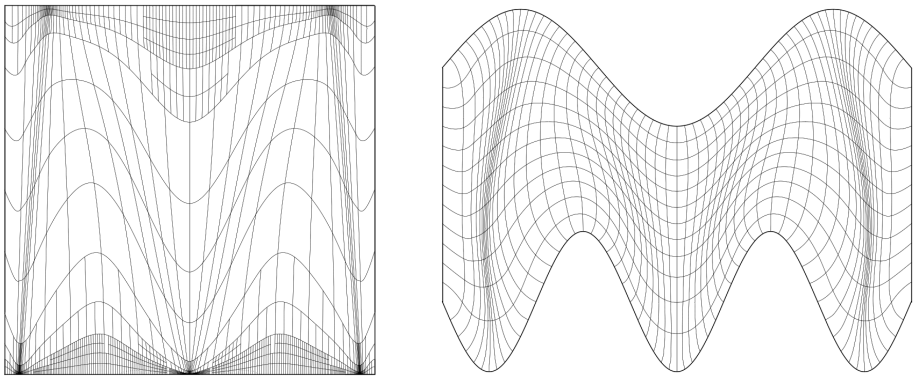
elements in the protruded parts compared to Figure 4.18. We introduce another control mapping $\mathbf{s}'(\boldsymbol{\xi})$, which we compute from the solution of

$$\int_{\hat{\Omega}} (\det \partial_{\mathbf{s}} \mathbf{x}_h)^k (g_{11} + \beta g_{22})_{\mathbf{s}\to\mathbf{s}'} \det \partial_{\boldsymbol{\xi}} \mathbf{s} \mathrm{d}S \to \min_{\mathbf{s}' \in V_h^2}, \quad \text{s.t.} \quad \mathbf{s}'(\boldsymbol{\xi}) = \mathbf{s}(\boldsymbol{\xi}) \text{ on } \partial\hat{\Omega}, \tag{4.84}$$

where $\mathbf{s} = (s,t)^T$ and $\mathbf{x}_h$ correspond to Figures 4.19 (a) and (b), respectively. Here, the g_{ii} correspond to diagonal entries of the metric tensor associated with the diffeomorphism between $\mathbf{s}|_{\hat{\Omega}}$ and $\mathbf{s}'|_{\hat{\Omega}}$. As before, $k > 0$ tunes to which degree the spread in cell size is penalized, while $\beta > 1$ tunes the degree to which $\mathbf{s}'$ is contracted / expanded in the direction of $\partial_\eta \mathbf{s}$, in order to compensate for large / small cells in $\mathbf{x}_h$. Taking β large essentially freezes $\mathbf{s}'$ in the direction of $\partial_\xi \mathbf{s}$, such that boundary orthogonality is preserved. Note that in (4.84), we are essentially solving the discrete counterpart of

$$\nabla_{\mathbf{s}} \cdot (D \nabla_{\mathbf{s}} \mathbf{s}'_i) = 0, \quad i \in \{1,2\}, \quad \text{s.t.} \quad \mathbf{s}'(\boldsymbol{\xi}) = \mathbf{s}(\boldsymbol{\xi}), \quad \text{with} \quad D = (\det \partial_{\mathbf{s}} \mathbf{x}_h)^k \begin{pmatrix} 1 & 0 \\ 0 & \beta \end{pmatrix}. \tag{4.85}$$

Figure 4.20 shows the geometry parameterization along with the reparameterized do-



(a) (b)

Figure 4.20: Result of reprameterizing the geometry mapping from Figure 4.19 using the principles from Section 4.4.1 (b) and the corresponding reparameterized domain (a).

main upon recomputation with $k = 0.75$ and $\beta = 300$. Compared to Figure 4.19, the figure shows a much better cell size distribution, in particular close to the boundaries. Large cells can be further penalized by increasing the value of k.

4.5. CHAPTER CONCLUSIONS

In this work, we presented a goal-oriented adaptive THB-spline framework for PDE-based planar parameterization. For this, we adopted the a posteriori refinement technique of dual weighted residual and proposed several goal-oriented refinement cost functions. This resulted in numerical schemes that combine iterative solution techniques with THB-enabled local a posteriori refinement strategies, hence avoiding over-refinement in computing a folding-free geometry parameterization.

In order to fine-tune the parametric properties of the resulting mapping, we combined aforementioned schemes with the concept of domain optimization. Hereby, the (convex) parametric domain, which constitutes the target domain of the mapping inverse, is reparameterized in order to alter the parametric properties of the recomputed mapping. For this, we proposed several optimization constraints that avoid the loss of bijectivity.

REFERENCES

[Aza09] Boris Nikolaevich Azarenok. Generation of structured difference grids in two-dimensional nonconvex domains using mappings. *Computational Mathematics and Mathematical Physics*, 49(5):797–809, 2009.

[BHW19] Jan Blechschmidt, Roland Herzog, and Max Winkler. Error estimation for second-order pdes in non-variational form. *arXiv preprint arXiv:1909.12676*, 2019.

[BMN01] Patricia Bauman, Antonella Marini, and Vincenzo Nesi. Univalent solutions of an elliptic system of partial differential equations arising in homogenization. *Indiana University Mathematics Journal*, pages 747–757, 2001.

[BZ09] Lorenz T Biegler and Victor M Zavala. Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575–582, 2009.

[CHB09] J Austin Cottrell, Thomas JR Hughes, and Yuri Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.

[FH99] Gerald Farin and Dianne Hansford. Discrete coons patches. *Computer Aided Geometric Design*, 16(7):691–700, 1999.

[FŠJ15] Antonella Falini, Jaka Špeh, and Bert Jüttler. Planar domain parameterization with THB-splines. *Computer Aided Geometric Design*, 35:95–108, 2015.

[Gal17] Dietmar Gallistl. Variational formulation and numerical analysis of linear elliptic equations in nondivergence form with cordes coefficients. *SIAM Journal on Numerical Analysis*, 55(2):737–757, 2017.

[GENN12] Jens Gravesen, Anton Evgrafov, Dang-Manh Nguyen, and Peter Nørtoft. Planar parametrization in isogeometric analysis. In *International Conference on Mathematical Methods for Curves and Surfaces*, pages 189–212. Springer, 2012.

[GJS12] Carlotta Giannelli, Bert Jüttler, and Hendrik Speleers. THB-splines: The truncated basis for hierarchical splines. *Computer Aided Geometric Design*, 29(7):485–498, 2012.

[HCB05] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *CMAME*, 194:4135–4195, 2005.

[HMV18a] J Hinz, M Möller, and C Vuik. Elliptic grid generation techniques in the framework of isogeometric analysis applications. *Computer Aided Geometric Design*, 2018.

[HMV18b] Jochen Hinz, Matthias Möller, and Cornelis Vuik. Spline-based parameterization techniques for twin-screw machine geometries. In *IOP Conference Series: Materials Science and Engineering*, volume 425, page 012030. IOP Publishing, 2018.

[KK98] Carl Timothy Kelley and David E Keyes. Convergence analysis of pseudo-transient continuation. *SIAM Journal on Numerical Analysis*, 35(2):508–523, 1998.

[KK04] Dana A Knoll and David E Keyes. Jacobian-free newton–krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[Kre91] E. Kreyszig. *Differential Geometry*. Differential Geometry. Dover Publications, 1991.

[LP11] Omar Lakkis and Tristan Pryer. A finite element method for second order nonvariational elliptic problems. *SIAM Journal on Scientific Computing*, 33(2):786–801, 2011.

[PO03] Serge Prudhomme and J Tinsley Oden. Computable error estimators and adaptive techniques for fluid flow problems. In *Error estimation and adaptive discretization methods in computational fluid dynamics*, pages 207–268. Springer, 2003.

[Ran04] Rolf Rannacher. Adaptive finite element methods in flow computations. *Recent Advances in Adaptive Computation. Contemporary Mathematics*, 383:183–176, 2004.

[RS07] Sergej Rjasanow and Olaf Steinbach. *The fast solution of boundary integral equations*. Springer Science & Business Media, 2007.

[SS10] Stefan A Sauter and Christoph Schwab. Boundary element methods. In *Boundary Element Methods*, pages 183–287. Springer, 2010.

[Ste93] Stanley Steinberg. *Fundamentals of grid generation*. CRC press, 1993.

[TSW98] Joe F Thompson, Bharat K Soni, and Nigel P Weatherill. *Handbook of grid generation*. CRC press, 1998.

[XMDG10] Gang Xu, Bernard Mourrain, Régis Duvigneau, and André Galligo. Optimal analysis-aware parameterization of computational domain in isogeometric analysis. In *International Conference on Geometric Modeling and Processing*, pages 236–254. Springer, 2010.

[XMDG11] Gang Xu, Bernard Mourrain, Régis Duvigneau, and André Galligo. Parameterization of computational domain in isogeometric analysis: methods and comparison. *Computer Methods in Applied Mechanics and Engineering*, 200(23-24):2021–2031, 2011.

5

THE ROLE OF PDE-BASED PARAMETERIZATION TECHNIQUES IN GRADIENT-BASED IGA SHAPE OPTIMIZATION APPLICATIONS

This chapter is based on the publication from [HJMV20]. It is motivated by the observation that the PDE-based formulation of the surface-to-volume problem $\partial\Omega \longrightarrow \Omega$ enables deriving a symbolic expression for the differential of the mapping with respect to the boundary correspondence $\partial\hat{\Omega} \rightarrow \partial\Omega$. This, in turn, leads to a relatively straightforward expression from which the differential of some objective function, which has a convoluted dependency on $\partial\Omega$, can be assembled. While a function evaluation may be relatively expensive, due to the need to solve a nonlinear problem for the mapping, computing the differential is cheap. Hence, optimization routines that employ approximate Hessian update strategies that require computing the function evaluation and its differential in tandem during each iteration, are especially well-suited for the proposed methodology.

This chapter proposes a shape optimization algorithm based on the principles of *Isogeometric Analysis* (IGA) in which the parameterization of the geometry enters the problem formulation as an additional PDE-constraint. Inspired by the *isoparametric principle* of IGA, the parameterization and the governing state equation are treated using the same numerical technique. This leads to a scheme that is comparatively easy to differentiate, allowing for a fully symbolic derivation of the gradient and subsequent gradient-based optimization. To improve the efficiency and ro-

bustness of the scheme, the basis is re-selected during each optimization iteration and adjusted to the current needs. The scheme is validated in two test cases.

5.1. INTRODUCTION

Isogeometric analysis (IGA) was introduced by Hughes et al. in [HCB05] as a numerical technique that bridges the gap between Computer Aided Design (CAD) and the numerical analysis of Partial Differential Equations (PDEs). This is accomplished by using the same function space to represent the geometry Ω and to discretize the PDE problem posed over Ω. Most of the available CAD software generates no more than a spline-based description of the boundary contours $\partial\Omega$ of Ω. Therefore, suitable parameterization algorithms are indispensable for generating bijective (folding-free), analysis-suitable geometry parameterizations from the boundary CAD data.

The parameteric quality of the mapping has a profound impact on the accuracy of the isogeometric analysis [XMDG10]. Therefore, besides bijectivity, proficient parameterization algorithms aim at generating parameterizations of high numerical quality.

A variety of parameterization techniques have been proposed in the literature such as Coon's Patch [FH99], Linear Spring [GENN12] and approaches based on (constrained and unconstrained) quality cost function optimization [GENN12, XMDG11, FŠJ15]. While mappings based on Coon's Patch and Linear Spring follow from a closed-form expression and are hence cheap to compute and straightforwardly differentiable, they often lead to folded (non-bijective) mappings. The same is true for unconstrained optimization. Constrained optimization approaches on the other hand typically have a higher success rate. However, this comes at the expense of a large number of (constrained) iterations (typically about ~ 30) and the notorieties associated with nonconvex optimization, such as the danger of getting stuck in local minima. A third class of approaches attempts to generate a mapping whose inverse is composed of harmonic functions in Ω. This approach is based on the observation that harmonic functions exhibit a large degree of smoothness, which benefits the numerical quality of the resulting mapping. Furthermore, it can be shown that inversely harmonic mappings (IHMs) are bijective, thanks to the maximum principle [Rad26, Kne26]. Many approaches for approximating IHMs have been proposed in the literature [NJ10, FŠJ15], notably the PDE-based approach called Elliptic Grid Generation (EGG) [Aza09, HMV18]. EGG is of particular interest in shape optimization problems thanks to the parametric smoothness and bijectivity of IHMs as well as differentiability, made possible by the PDE-based problem formulation.

Traditionally, IGA parameterizations are taken from tensor-product spline spaces. Unfortunately, structured spline technologies do not allow for local refinement. This may result in infeasibly-large function spaces. Therefore, unstructured spline technologies such as THB-splines [GJS12] are gaining an increased amount interest in the IGA community, thanks to local refinement. An EGG-based planar parameterization framework that supports THB-splines has been proposed in [HAM20] (see Chapter 4).

Since its birth in 2005, IGA has been successfully applied to wide variety of problems including: thermal analysis [KOŁ14], linear elasticity problems [HCB05], structural vibrations [CRBH06], incompressible flows [BH08] and inviscid compressible flows [Jae15]. As a mature numerical method, it is ready to be used in more complex industrial pro-

cesses. As a result, several publications that apply IGA to shape optimization problems have appeared in the literature [YHC13, NG13, WFC08, MEGG11]. Combining IGA and shape optimization is very appealing as the spline-based description of $\partial\Omega$ can be used directly to compute a mapping for Ω, completely bypassing the need to first convert $\partial\Omega$ into a piecewise-linear curve that acts as an input for classical mesh generators.
There are two main groups of shape optimization algorithms: gradient-free (like for example genetic algorithms [Hol84]) and gradient-based methods (for example interior point methods [WB06, BZ09]). The latter group generally requires fewer underlying PDE evaluations at the expense of having to compute the gradient of the objective function during each iteration. Therefore, differentiability of the IGA parameterization algorithm constitutes a significant advantage. An additional feature of differentiability is efficiency: as the inner control points are a smooth function of the boundary control points, there is no need for full remeshing after each iteration since cheaper mesh update strategies can be employed. This is also true for settings in which the boundary contours change as a smooth function of time.
In order to combine the appealing features of EGG and THB-enabled local refinement, this Chapter adopts the parameterization framework proposed in [HAM20] (see Chapter 4) and presents an IGA-based shape optimization algorithm in which the parameterization is added to the optimization problem formulation in the form of an additional PDE-constraint. In line with the *isoparametric principle* of IGA, we numerically treat this additional constraint in the same way as the governing quantity (temperature, pressure, etc) of the underlying optimization problem. Including the mapping explicitly as a PDE-constraint facilitates differentiation, allowing for gradient-based optimization, while also guaranteeing analysis-suitability, thanks to the bijectivity of IHMs. To improve the efficiency, the proposed algorithm employs THB-enabled adaptive local refinement strategies during every optimization iteration, resulting in a variable discretization basis. We validate the proposed methodology by presenting two test cases.

5.2. CHAPTER NOTATION

In this chapter, we denote vectors in boldface while matrices receive a capital letter and may furthermore be enclosed in square brackets for better readability. The i-th entry of vector $\mathbf{x}$ is denoted by $\mathbf{x}_i$ or simply x_i and similarly for the ij-th entry of matrices. We make extensive use of vector derivatives. Here, we interchangeably use the denotation

$$[\partial_{\mathbf{t}}\mathbf{x}] \equiv \left[\frac{\partial \mathbf{x}}{\partial \mathbf{t}}\right], \quad \text{with} \quad \left[\frac{\partial \mathbf{x}}{\partial \mathbf{t}}\right]_{ij} = \frac{\partial x_i}{\partial t_j} \tag{5.1}$$

for the partial derivative and similarly for the total derivative. In the case of taking the derivative of a scalar, brackets are avoided. However, the argument is treated as a 1×1 matrix and hence the derivative has dimension $(1, m)$, where m is the dimension of $\mathbf{t}$. When integrating locally defined quantities $u(\boldsymbol{\xi}) : \hat{\Omega} \to \mathbb{R}^n$ over the physical domain Ω, we avoid mentioning the push-forward with the mapping $\mathbf{x} : \hat{\Omega} \to \Omega$ for convenience, i.e.,

$$\int_{\Omega} u \circ \mathbf{x}^{-1} \mathrm{d}S = \int_{\hat{\Omega}} u(\boldsymbol{\xi}) \det\left[\partial_{\boldsymbol{\xi}}\mathbf{x}\right] \mathrm{d}\boldsymbol{\xi} \longrightarrow \int_{\Omega} u(\mathbf{x}) \mathrm{d}S, \tag{5.2}$$

and assume that the reader is aware of the mathematical subtleties involved.

5.3. PROBLEM FORMULATION

We are considering the shape optimization problem of a planar domain $\Omega(\boldsymbol{\alpha})$ whose contours $\partial\Omega(\boldsymbol{\alpha})$ are parameterized by the n-tuple of design variables $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)$. If the design variables are taken from the design space $\boldsymbol{\lambda}$, the optimization problem reads:

$$\begin{aligned} & J\left(u^{\boldsymbol{\alpha}}, \Omega^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}\right) \to \min_{\boldsymbol{\alpha}} \\ \text{s.t.} \quad & g_i(u^{\boldsymbol{\alpha}}, \Omega^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \geq 0, \quad \forall i \in \{1, \dots, N_{\neq}\} \\ & h_j(u^{\boldsymbol{\alpha}}, \Omega^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) = 0, \quad \forall j \in \{1, \dots, N_{=}\} \\ & \boldsymbol{\alpha} \in \boldsymbol{\lambda}, \end{aligned} \tag{5.3}$$

where the g_i and h_j are problem-specific constraints. Here, $J(\cdot,\cdot,\cdot)$ denotes the objective function and $u^{\boldsymbol{\alpha}} : \Omega \to \mathbb{R}$ some state variable whose physical meaning depends on the application (temperature, pressure, etc). We regard $u^{\boldsymbol{\alpha}}$ as a scalar quantity for convenience. However, generalizations to vectorial quantities are straightforward. Note that the dependencies of the variables contained in $J(\cdot,\cdot,\cdot)$ are concatenated in descending order, i.e., in general $u^{\boldsymbol{\alpha}} = u^{\boldsymbol{\alpha}}(\Omega^{\boldsymbol{\alpha}}(\boldsymbol{\alpha}), \boldsymbol{\alpha})$ and $\Omega^{\boldsymbol{\alpha}} = \Omega^{\boldsymbol{\alpha}}(\boldsymbol{\alpha})$. The state variable $u^{\boldsymbol{\alpha}}$ follows from a PDE-problem posed over $\Omega^{\boldsymbol{\alpha}}$ and may contain additional dependencies on $\boldsymbol{\alpha}$ (such as source terms), hence the dependency on the tuple $(\Omega^{\boldsymbol{\alpha}}, \boldsymbol{\alpha})$. Tackling (5.3) computationally requires introducing a bijective geometry parameterization $\mathbf{x}^{\boldsymbol{\alpha}} : \hat{\Omega} \to \Omega^{\boldsymbol{\alpha}}$, where $\hat{\Omega}$ denotes the parametric domain which is assumed to be static. Here, we restrict ourselves to geometries that are topologically equivalent to $\hat{\Omega} = (0,1)^2$ for convenience. However, the generalization to multipatch settings is straightforward. Let

$$\mathscr{U}^f = \{v \in \mathscr{U} \mid v = f \text{ on } \partial\Omega_D^{\boldsymbol{\alpha}}\} \tag{5.4}$$

for some suitably-chosen vector space $\mathscr{U}$ and some $\partial\Omega_D^{\boldsymbol{\alpha}} \subseteq \partial\Omega^{\boldsymbol{\alpha}}$ on which Dirichlet data is prescribed. Deriving the weak form of the PDE-problem governing $u^{\boldsymbol{\alpha}}$ leads to

$$\text{find } u^{\boldsymbol{\alpha}} \in \mathscr{U}^{u_D^{\boldsymbol{\alpha}}} \quad \text{s.t.} \quad B\left(u^{\boldsymbol{\alpha}}, \mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}, \phi\right) = 0, \quad \forall \phi \in \mathscr{U}^0, \tag{5.5}$$

for some differential form $B(\cdot,\cdot,\cdot,\cdot)$. Here, $u_D^{\boldsymbol{\alpha}}$ denotes the Dirichlet data as a function of the design variables. By introducing the mapping $\mathbf{x}^{\boldsymbol{\alpha}}$, the objective function takes the form

$$J(u^{\boldsymbol{\alpha}}, \Omega^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \to J\left(u^{\boldsymbol{\alpha}}, \mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}\right) \equiv J^{\boldsymbol{\alpha}}, \tag{5.6}$$

where $u^{\boldsymbol{\alpha}}$ satisfies (5.5). With the dependencies of $u^{\boldsymbol{\alpha}}$ and $\mathbf{x}^{\boldsymbol{\alpha}}$ in mind, the gradient of (5.6) reads:

$$\frac{\mathrm{d}J^{\boldsymbol{\alpha}}}{\mathrm{d}\boldsymbol{\alpha}} = \frac{\partial J^{\boldsymbol{\alpha}}}{\partial u^{\boldsymbol{\alpha}}}\left(\frac{\partial u^{\boldsymbol{\alpha}}}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}\frac{\mathrm{d}\mathbf{x}^{\boldsymbol{\alpha}}}{\mathrm{d}\boldsymbol{\alpha}} + \frac{\partial u^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}}\right) + \frac{\partial J^{\boldsymbol{\alpha}}}{\partial \mathbf{x}^{\boldsymbol{\alpha}}}\frac{\mathrm{d}\mathbf{x}^{\boldsymbol{\alpha}}}{\mathrm{d}\boldsymbol{\alpha}} + \frac{\partial J^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}}. \tag{5.7}$$

We see that (5.7) requires taking the derivative of $\mathbf{x}^{\boldsymbol{\alpha}}$ with respect to $\boldsymbol{\alpha}$, while the state variable $u^{\boldsymbol{\alpha}}$ needs to be differentiable with respect to $\mathbf{x}^{\boldsymbol{\alpha}}$. These two derivatives often constitute the most challenging step in computing the gradient because differentiating $\mathbf{x}^{\boldsymbol{\alpha}}$ or with respect to $\mathbf{x}^{\boldsymbol{\alpha}}$ can be nontrivial, depending on the parameterization technique used. On the other hand, differentiation with respect to $u^{\boldsymbol{\alpha}}$ is relatively straightforward

because the implicit function theorem can be used on (5.5). Hence, if we take $\mathbf{x}^{\boldsymbol{\alpha}}$ as the solution of a PDE problem, differentiation is simplified, allowing for a symbolic derivation of all terms involved in (5.7). To this end, we adopt the principles of *Elliptic Grid Generation*, which will be the topic of the next section.

5.4. ELLIPTIC GRID GENERATION

Elliptic grid generation (EGG) is a PDE-based technique aimed at generating analysis-suitable geometry parameterizations $\mathbf{x}^{\boldsymbol{\alpha}} : \hat{\Omega} \to \Omega^{\boldsymbol{\alpha}}$ given only a parametric description of the boundary contours $\partial\Omega^{\boldsymbol{\alpha}}$ as a function of the state vector $\boldsymbol{\alpha}$. Let the free topological variables in $\hat{\Omega}$ be given by the tuple $\boldsymbol{\xi} = (\xi_1, \xi_2)^T = (\xi, \eta)^T$. Then, the equations of EGG read [HAM20]:

$$A(\mathbf{x}^{\boldsymbol{\alpha}}) : H(\mathbf{x}_i^{\boldsymbol{\alpha}}) = 0 \quad \text{in } \hat{\Omega}, \quad \text{for } i \in \{1,2\} \quad \text{s.t.} \quad \mathbf{x}^{\boldsymbol{\alpha}}|_{\partial\hat{\Omega}} = \partial\Omega^{\boldsymbol{\alpha}}, \tag{5.8}$$

where

$$H(u)_{ij} \equiv \frac{\partial^2 u}{\partial\xi_i \partial\xi_j} \quad \text{and} \quad A(\mathbf{x}^{\boldsymbol{\alpha}}) = \frac{1}{g_{11} + g_{22} + \epsilon} \begin{pmatrix} g_{22} & -g_{12} \\ -g_{12} & g_{11} \end{pmatrix}, \tag{5.9}$$

with $g_{ij} = \mathbf{x}^{\boldsymbol{\alpha}}_{\xi_i} \cdot \mathbf{x}^{\boldsymbol{\alpha}}_{\xi_j}$ the entries of the metric tensor and ϵ a small positive constant (typically, we take $\epsilon = 10^{-4}$). Here, $A : B$ denotes the Frobenius inner product between matrices A and B. The solution of (5.8) is a mapping $\mathbf{x}^{\boldsymbol{\alpha}}$ whose inverse $(\mathbf{x}^{\boldsymbol{\alpha}})^{-1}$ constitutes a pair of harmonic functions on $\Omega^{\boldsymbol{\alpha}}$. As $(\mathbf{x}^{\boldsymbol{\alpha}})^{-1}$ maps into a convex parametric domain $\hat{\Omega}$, it follows from the maximum principle that $\mathbf{x}^{\boldsymbol{\alpha}}$ is a bijection between $\hat{\Omega}$ and $\Omega^{\boldsymbol{\alpha}}$, where $\mathbf{x}^{\boldsymbol{\alpha}}|_{\partial\hat{\Omega}}$ parameterizes $\partial\Omega^{\boldsymbol{\alpha}}$ [Rad26, Kne26]. This property justifies limiting the choice of $\mathbf{x}^{\boldsymbol{\alpha}}$ from the set of bijective parameterizations to the subset of mappings that satisfy (5.8). While many alternative approaches that do not require solving a PDE problem have been proposed in the literature [NJ10, FŠJ15], we base a computational approach on (5.8) since it facilitates differentiating $\mathbf{x}^{\boldsymbol{\alpha}}$ with respect to the design variables (see (5.7)).

For a viable computational approach, we derive the weak counterpart of (5.8). Here, we adopt the approach from [HMV18]. Given a differential function $\mathbf{x}_D^{\boldsymbol{\alpha}} : \partial\hat{\Omega} \to \mathbb{R}^2$ that parameterizes $\partial\Omega^{\boldsymbol{\alpha}}$, the mapping $\mathbf{x}^{\boldsymbol{\alpha}}$ is the solution of:

$$\text{find } \mathbf{x}^{\boldsymbol{\alpha}} \in \boldsymbol{V}^{\mathbf{x}_D^{\boldsymbol{\alpha}}} \quad \text{s.t.} \quad F(\mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\sigma}) = 0, \quad \forall \boldsymbol{\sigma} \in \boldsymbol{V}^{\mathbf{0}}, \tag{5.10}$$

with

$$F(\mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\sigma}) = \sum_{i=1}^{2} \int_{\hat{\Omega}} \boldsymbol{\sigma}_i A(\mathbf{x}^{\boldsymbol{\alpha}}) : H(\mathbf{x}_i^{\boldsymbol{\alpha}}) \mathrm{d}S. \tag{5.11}$$

In (5.10), we used

$$\boldsymbol{V}^{\mathbf{f}} \equiv \{\mathbf{v} \in \mathcal{V}^2 \mid \mathbf{v} = \mathbf{f} \text{ on } \partial\hat{\Omega}\}, \quad \text{with} \quad \mathcal{V} = H^2(\hat{\Omega}) \quad \text{and} \quad \mathbf{f} \in \mathcal{V}^2. \tag{5.12}$$

The discretization of (5.10) follows straightforwardly from replacing $\mathcal{V}$ by the finite-dimensional $\mathcal{V}_h^{\boldsymbol{\alpha}} \subset \mathcal{V}$ in (5.12). We denote the resulting set by $\boldsymbol{V}_h^{\boldsymbol{\alpha},\mathbf{f}}$. As $\mathbf{f} \in \mathcal{V}_h^{\boldsymbol{\alpha}} \times \mathcal{V}_h^{\boldsymbol{\alpha}} \equiv \boldsymbol{V}_h^{\boldsymbol{\alpha}}$ by assumption, the discretization additionally requires replacing the Dirichlet data $\mathbf{x}_D^{\boldsymbol{\alpha}}$ by a proper collocation $\mathbf{x}_{D,h}^{\boldsymbol{\alpha}} \in \boldsymbol{V}_h^{\boldsymbol{\alpha}} \setminus \boldsymbol{V}_h^{\boldsymbol{\alpha},\mathbf{0}}$. As such, the fully discretized problem reads:

$$\text{find } \mathbf{x}_h^{\boldsymbol{\alpha}} \in \boldsymbol{V}^{\boldsymbol{\alpha},\mathbf{x}_{D,h}^{\boldsymbol{\alpha}}} \quad \text{s.t.} \quad F(\mathbf{x}_h^{\boldsymbol{\alpha}}, \boldsymbol{\sigma}_h) = 0, \quad \forall \boldsymbol{\sigma}_h \in \boldsymbol{V}_h^{\boldsymbol{\alpha},\mathbf{0}}. \tag{5.13}$$

Remark. Due to the appearance of second order derivatives in (5.13), we have to assume that $\mathbf{x}_h^{\boldsymbol{\alpha}}$ is built from a space with global $C^1(\hat{\Omega})$-continuity. For an approach that allows for lower regularity (and is hence compatible with multipatch parameterizations), we refer to [HMV19] (see Chapter 3).

Since (5.13) is a nonlinear root-finding problem, we tackle it with a Newton-based iterative approach. Unlike $\mathbf{x}^{\boldsymbol{\alpha}}$, its discretized counterpart $\mathbf{x}_h^{\boldsymbol{\alpha}}$ may fold due to the truncation error introduced by the numerical scheme. Grid folding can be repaired by refining $\mathcal{V}_h^{\boldsymbol{\alpha}}$ in the affected regions and recomputing $\mathbf{x}_h^{\boldsymbol{\alpha}}$ from the enriched space. This makes using an unstructured spline technology like THB-splines particularly appealing, thanks to local refinement. For more details on the choice of $\mathcal{V}_h^{\boldsymbol{\alpha}}$ and the algorithm that tackles the nonlinear root-finding problem (5.13), we refer to [HAM20] (see Chapter 4).

5.5. COMPUTATIONAL APPROACH

In this section we propose a computational approach for numerically treating the optimization problem (5.3).

5.5.1. DISCRETIZATION

We discretize the optimization problem (5.3) by approximating

$$J(u^{\boldsymbol{\alpha}}, \mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \simeq J(u_h^{\boldsymbol{\alpha}}, \mathbf{x}_h^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \equiv J_h^{\boldsymbol{\alpha}}, \tag{5.14}$$

where $u_h^{\boldsymbol{\alpha}} \in \mathcal{U}_h^{\boldsymbol{\alpha}}$ is the solution of the discretized weak state equation (5.5) while $\mathbf{x}_h^{\boldsymbol{\alpha}} \in \boldsymbol{V}_h^{\boldsymbol{\alpha}}$ is the solution of (5.13) for given $\boldsymbol{\alpha}$. Here, $\Omega_h^{\boldsymbol{\alpha}}$ is parameterized by $\mathbf{x}_h^{\boldsymbol{\alpha}}$ and approximates the domain $\Omega^{\boldsymbol{\alpha}}$ whose contours are parameterized by the $\boldsymbol{\alpha}$-differentiable $\mathbf{x}_D^{\boldsymbol{\alpha}} : \partial\hat{\Omega} \to \mathbb{R}^2$ which we consider a given function. The distance

$$D(\partial\Omega_h^{\boldsymbol{\alpha}}, \partial\Omega^{\boldsymbol{\alpha}}) \equiv \left\| \mathbf{x}_{D,h}^{\boldsymbol{\alpha}} - \mathbf{x}_D^{\boldsymbol{\alpha}} \right\|_{L_2(\partial\hat{\Omega})} \tag{5.15}$$

serves as a measure of the approximation quality.
Likewise, we approximate the gradient by replacing $(u^{\boldsymbol{\alpha}}, \mathbf{x}^{\boldsymbol{\alpha}}) \to (u_h^{\boldsymbol{\alpha}}, \mathbf{x}_h^{\boldsymbol{\alpha}})$ in (5.7), i.e.,

$$\frac{\mathrm{d}J}{\mathrm{d}\boldsymbol{\alpha}} \simeq \frac{\partial J_h^{\boldsymbol{\alpha}}}{\partial u_h^{\boldsymbol{\alpha}}} \left(\frac{\partial u_h^{\boldsymbol{\alpha}}}{\partial \mathbf{x}_h^{\boldsymbol{\alpha}}} \frac{\mathrm{d}\mathbf{x}_h^{\boldsymbol{\alpha}}}{\mathrm{d}\boldsymbol{\alpha}} + \frac{\partial u_h^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}} \right) + \frac{\partial J_h^{\boldsymbol{\alpha}}}{\partial \mathbf{x}_h^{\boldsymbol{\alpha}}} \frac{\mathrm{d}\mathbf{x}_h^{\boldsymbol{\alpha}}}{\mathrm{d}\boldsymbol{\alpha}} + \frac{\partial J_h^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}}. \tag{5.16}$$

At this point, it should be noted that for given $\boldsymbol{\alpha}$, the exact evaluations of $J(\cdot,\cdot,\cdot)$ and the components of its gradient are independent of the particular choice of the coordinate system $\mathbf{x}^{\boldsymbol{\alpha}}$. As such, the quality of the approximations introduced in (5.14) and (5.16) depend solely on the numerical accuracy of $u_h^{\boldsymbol{\alpha}}$, which in turn is affected by the parametric quality of $\mathbf{x}_h^{\boldsymbol{\alpha}}$ and the distance of $\partial\Omega_h^{\boldsymbol{\alpha}}$ to the exact $\partial\Omega^{\boldsymbol{\alpha}}$.
We numerically treat (5.3) based on a *variable basis approach* (VBA) rather than a *static basis approach* (SBA). In SBA, $u_h^{\boldsymbol{\alpha}}$ and $\mathbf{x}_h^{\boldsymbol{\alpha}}$ are constructed from the static tuple $(\mathcal{U}_h, \boldsymbol{V}_h)$, while in VBA the tuple $(\mathcal{U}_h^{\boldsymbol{\alpha}}, \boldsymbol{V}_h^{\boldsymbol{\alpha}})$ may be chosen differently during each iteration and is tuned to the current needs. We make a choice based on the following principles:

A. $\left\| \mathbf{x}_{D,h}^{\boldsymbol{\alpha}} - \mathbf{x}_D^{\boldsymbol{\alpha}} \right\|$, with $\mathbf{x}_{D,h}^{\boldsymbol{\alpha}} \in \boldsymbol{V}_h^{\boldsymbol{\alpha}} \setminus \boldsymbol{V}_h^{\boldsymbol{\alpha},\mathbf{0}}$ is sufficiently small;

B. $\mathbf{x}_h^{\boldsymbol{\alpha}} \in \boldsymbol{V}_h^{\boldsymbol{\alpha}}$, resulting from $\mathbf{x}_{D,h}^{\boldsymbol{\alpha}}$ in combination with (5.13), is a bijection and preferably of high numerical quality;

C. $u_h^{\boldsymbol{\alpha}} \in \mathscr{U}_h^{\boldsymbol{\alpha}}$ approximates $u^{\boldsymbol{\alpha}}$ well.

As such, for given $\boldsymbol{\alpha}$, we select the tuple $(\mathscr{U}_h^{\boldsymbol{\alpha}}, \boldsymbol{V}_h^{\boldsymbol{\alpha}})$ such that points A to C are satisfied with a minimal number of degrees of freedom (DOFs).
In SBA, a necessary condition for local optimality follows straightforwardly from the discretized counterpart of (5.3) over the static tuple $(\mathscr{U}_h, \mathcal{V}_h)$. In contrast, VBA necessitates basing such a condition on (5.3) *before* discretization. Hence, numerical assessment of local optimality in (5.3) is obligatory, due to the approximate nature of $J_h^{\boldsymbol{\alpha}}$ and its gradient. This may be regarded as a drawback since it can generate false positives / negatives caused by the truncation error at the current iterate. On the other hand, VBA allows for $\boldsymbol{\alpha}$-specific feature-based basis selection for approximating both $\mathbf{x}^{\boldsymbol{\alpha}}$ and $u^{\boldsymbol{\alpha}}$, leading to a highly flexible scheme. When performing shape optimization in combination with EGG, a static $\boldsymbol{V}_h$ may be inappropriate for particular choices of $\boldsymbol{\alpha}$ which results in grid-folding (impeding the evaluation of $J_h^{\boldsymbol{\alpha}}$), hence justifying VBA-enabled feature-based basis selection in applications which are geometrically complex.

Remark. If we regard the truncation error $\tau(\boldsymbol{\alpha})$ in $u^{\boldsymbol{\alpha}} = u_h^{\boldsymbol{\alpha}} + \tau(\boldsymbol{\alpha})$ as a random variable drawn from some probability distribution, above methodology possesses many properties reminiscent of stochastic gradient descent [Bot10]. As such, the convergence tolerance should be designed with the expected magnitude of $\tau(\boldsymbol{\alpha})$ (and its contribution to the gradient) in mind and hence taken generously. Here, we regard this as a minor shortcoming since we consider complex and highly nonconvex, nonlinear optimization problems in which the model error as well as the notorieties associated with nonconvex optimization (such as the danger of getting stuck in local minima) pose a greater threat to solution quality than the truncation error in practice. Furthermore, in most practical applications, a particular state vector need not be optimal in order to be considered adequate.

5.5.2. GRADIENT-BASED OPTIMIZATION USING AN ADJOINT FORMULATION

In the following, we present a scheme that is suitable for gradient-based optimization, where all terms involved are assembled from expressions that have been derived fully symbolically. For given $\boldsymbol{\alpha}$, we assume that a suitable tuple $(\mathscr{U}_h^{\boldsymbol{\alpha}}, \boldsymbol{V}_h^{\boldsymbol{\alpha}})$ has been chosen based on principles A to C (see section 5.5.1). Particular methodologies for satisfying these principles depend on the application and are discussed in Section 5.6. In the following, the operator $[\cdot]$ returns the canonical basis of a vector space, which we assume to be clear from context. Reminiscent of (5.12), we introduce

$$\boldsymbol{V}_h^{\boldsymbol{\alpha},\mathbf{f}} \equiv \{\mathbf{v} \in \boldsymbol{V}_h^{\boldsymbol{\alpha}} \mid \mathbf{v} = \mathbf{f} \text{ on } \partial\hat{\Omega}\} \quad \text{and} \quad \mathscr{U}_h^{\boldsymbol{\alpha},f} \equiv \{v \in \mathscr{U}_h^{\boldsymbol{\alpha}} \mid v = f \text{ on } \partial\hat{\Omega}_D\}, \tag{5.17}$$

where, as before, $\mathbf{f} \in \boldsymbol{V}_h^{\boldsymbol{\alpha}}$ and $f \in \mathscr{U}_h^{\boldsymbol{\alpha}}$ by assumption. In (5.17), $\partial\hat{\Omega}_D \subseteq \partial\hat{\Omega}$ refers to the preimage of $\partial\Omega_{D,h}^{\boldsymbol{\alpha}} \subseteq \partial\Omega_h^{\boldsymbol{\alpha}}$ under the mapping $\mathbf{x}_h^{\boldsymbol{\alpha}}$. As opposed to (5.4), here $\mathscr{U}_h^{\boldsymbol{\alpha}}$ is defined in the static parametric domain for convenience. Equation (5.17) allows for the decomposition into *boundary* ($\mathscr{B}$) and *inner* ($\mathscr{I}$) bases:

$$[\boldsymbol{V}_h^{\boldsymbol{\alpha}}] = \left[\boldsymbol{V}_h^{\boldsymbol{\alpha},\mathscr{B}}\right] \cup \left[\boldsymbol{V}_h^{\boldsymbol{\alpha},\mathscr{I}}\right], \quad \text{with} \quad \boldsymbol{V}_h^{\boldsymbol{\alpha},\mathscr{I}} = \boldsymbol{V}_h^{\boldsymbol{\alpha},\mathbf{0}} \quad \text{and} \quad \boldsymbol{V}_h^{\boldsymbol{\alpha},\mathscr{B}} = \boldsymbol{V}_h^{\boldsymbol{\alpha}} \setminus \boldsymbol{V}_h^{\boldsymbol{\alpha},\mathscr{I}}, \tag{5.18}$$

and similarly for $\mathscr{U}_h^{\boldsymbol{\alpha}}$. Given $\mathbf{x}_{D,h}^{\boldsymbol{\alpha}} \in \boldsymbol{V}_h^{\boldsymbol{\alpha},\mathscr{B}}$ (see Section 5.5.1), we introduce the mapping

$$\mathbf{x}_h^{\boldsymbol{\alpha}} = \mathbf{x}_0^{\boldsymbol{\alpha}} + \mathbf{x}_{D,h}^{\boldsymbol{\alpha}}, \quad \text{with} \quad \mathbf{x}_0^{\boldsymbol{\alpha}} = \sum_{\boldsymbol{\sigma}_i \in \left[\boldsymbol{V}_h^{\boldsymbol{\alpha},\mathscr{I}}\right]} c_i^{\mathscr{I}} \boldsymbol{\sigma}_i \quad \text{and} \quad \mathbf{x}_{D,h}^{\boldsymbol{\alpha}} = \sum_{\boldsymbol{\sigma}_j \in \left[\boldsymbol{V}_h^{\boldsymbol{\alpha},\mathscr{B}}\right]} c_j^{\mathscr{B}} \boldsymbol{\sigma}_j, \tag{5.19}$$

where the $c_j^{\mathscr{B}}$ are known. We introduce the vector of weights $\mathbf{c}_{\mathscr{A}}$ (where the subscript $\mathscr{A}$ stands for *all*), which is the concatenation of the vectors $\mathbf{c}_{\mathscr{I}}$ and $\mathbf{c}_{\mathscr{B}}(\boldsymbol{\alpha})$, containing the $c_i^{\mathscr{I}}$ and $c_j^{\mathscr{B}}$, respectively. Similarly, we introduce

$$u_h^{\boldsymbol{\alpha}} = \sum_{\phi_i \in \left[\mathscr{U}_h^{\boldsymbol{\alpha}}\right]} d_i \phi_i \quad \text{with the corresponding vector of weights} \quad (\mathbf{d}_{\mathscr{A}})_i = d_i. \tag{5.20}$$

With the introduction of the tuple $(\mathbf{c}_{\mathscr{A}}, \mathbf{d}_{\mathscr{A}})$, the discrete objective function is rewritten in the form

$$J_h^{\boldsymbol{\alpha}}(\mathbf{x}_h^{\boldsymbol{\alpha}}, u_h^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \longrightarrow J_h^{\boldsymbol{\alpha}}(\mathbf{c}_{\mathscr{A}}, \mathbf{d}_{\mathscr{A}}, \boldsymbol{\alpha}), \tag{5.21}$$

with

$$\mathbf{c}_{\mathscr{A}} = \mathbf{c}_{\mathscr{A}}(\mathbf{c}_{\mathscr{I}}, \mathbf{c}_{\mathscr{B}}), \quad \mathbf{c}_{\mathscr{I}} = \mathbf{c}_{\mathscr{I}}(\mathbf{c}_{\mathscr{B}}), \quad \mathbf{c}_{\mathscr{B}} = \mathbf{c}_{\mathscr{B}}(\boldsymbol{\alpha}) \quad \text{and} \quad \mathbf{d}_{\mathscr{A}} = \mathbf{d}_{\mathscr{A}}(\mathbf{c}_{\mathscr{A}}, \boldsymbol{\alpha}). \tag{5.22}$$

With above concatenated dependencies in mind, the transposed gradient approximation reads:

$$\frac{\mathrm{d}J_h^{\boldsymbol{\alpha}}}{\mathrm{d}\boldsymbol{\alpha}}^T = \left[\frac{\mathrm{d}\mathbf{c}_{\mathscr{A}}}{\mathrm{d}\boldsymbol{\alpha}}\right]^T \left(\left[\frac{\partial \mathbf{d}_{\mathscr{A}}}{\partial \mathbf{c}_{\mathscr{A}}}\right]^T \frac{\partial J_h^{\boldsymbol{\alpha}}}{\partial \mathbf{d}_{\mathscr{A}}}^T + \frac{\partial J_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathscr{A}}}^T \right) + \left[\frac{\mathrm{d}\mathbf{d}_{\mathscr{A}}}{\mathrm{d}\boldsymbol{\alpha}}\right]^T \frac{\partial J_h^{\boldsymbol{\alpha}}}{\partial \mathbf{d}_{\mathscr{A}}}^T + \frac{\partial J_h^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}}^T, \tag{5.23}$$

where we denoted matrix quantities in square brackets.
Introducing the discrete EGG residual vector $\mathbf{F}_h^{\boldsymbol{\alpha}}$ with

$$\left(\mathbf{F}_h^{\boldsymbol{\alpha}}\right)_i = F(\mathbf{x}_h^{\boldsymbol{\alpha}}, \boldsymbol{\sigma}_i), \quad \text{for} \quad \boldsymbol{\sigma}_i \in \left[\boldsymbol{V}_h^{\boldsymbol{\alpha},\mathscr{I}}\right] \quad \text{and} \quad F(\cdot,\cdot) \text{ as defined in (5.11)}, \tag{5.24}$$

we use the implicit function theorem [KP12] to derive an expression for the gradient of $\mathbf{c}_{\mathscr{A}}$. We have:

$$\left[\frac{\mathrm{d}\mathbf{c}_{\mathscr{A}}}{\mathrm{d}\boldsymbol{\alpha}}\right]^T = \left[\left[\frac{\mathrm{d}\mathbf{c}_{\mathscr{I}}}{\mathrm{d}\boldsymbol{\alpha}}\right]^T, \left[\frac{\partial \mathbf{c}_{\mathscr{B}}}{\partial \boldsymbol{\alpha}}\right]^T\right], \quad \text{with} \quad \left[\frac{\mathrm{d}\mathbf{c}_{\mathscr{I}}}{\mathrm{d}\boldsymbol{\alpha}}\right] = -\left[\frac{\partial \mathbf{F}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathscr{I}}}\right]^{-1} \left[\frac{\partial \mathbf{F}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathscr{B}}}\right] \left[\frac{\partial \mathbf{c}_{\mathscr{B}}}{\partial \boldsymbol{\alpha}}\right]. \tag{5.25}$$

Similarly, we define the residual vector $\mathbf{B}_h^{\boldsymbol{\alpha}}$ of the discretized weak state equation (see (5.5)), with entries

$$\left(\mathbf{B}_h^{\boldsymbol{\alpha}}\right)_i = B\left(u_h^{\boldsymbol{\alpha}}, \mathbf{x}_h^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}, \phi_i\right), \quad \text{for} \quad \phi_i \in \left[\mathscr{U}_h^{\boldsymbol{\alpha},\mathscr{I}}\right]. \tag{5.26}$$

The implicit function theorem yields

$$\left[\frac{\partial \mathbf{d}_{\mathscr{A}}}{\partial \mathbf{c}_{\mathscr{A}}}\right] = -\left[\frac{\partial \mathbf{B}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{d}_{\mathscr{A}}}\right]^{-1} \left[\frac{\partial \mathbf{B}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathscr{A}}}\right] \quad \text{and} \quad \left[\frac{\mathrm{d}\mathbf{d}_{\mathscr{A}}}{\mathrm{d}\boldsymbol{\alpha}}\right] = -\left[\frac{\partial \mathbf{B}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{d}_{\mathscr{A}}}\right]^{-1} \left[\frac{\partial \mathbf{B}_h^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}}\right]. \tag{5.27}$$

Substituting in (5.23) leads to

$$\frac{\mathrm{d}J_h^{\boldsymbol{\alpha}}}{\mathrm{d}\boldsymbol{\alpha}}^T = \left[\frac{\mathrm{d}\mathbf{c}_{\mathscr{A}}}{\mathrm{d}\boldsymbol{\alpha}}\right]^T \mathbf{b} - \left[\frac{\partial \mathbf{B}_h^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}}\right]^T \mathbf{a} + \frac{\partial J_h^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}}^T, \tag{5.28}$$

with

$$\mathbf{a} = \left[\frac{\partial \mathbf{B}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{d}_{\mathscr{A}}}\right]^{-T} \frac{\partial J_h^{\boldsymbol{\alpha}}}{\partial \mathbf{d}_{\mathscr{A}}}^T \quad \text{and} \quad \mathbf{b} = -\left[\frac{\partial \mathbf{B}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathscr{A}}}\right]^T \mathbf{a} + \frac{\partial J_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathscr{A}}}^T. \tag{5.29}$$

Vector $\mathbf{a}$ is computed by solving the following linear system:

$$\left[\frac{\partial \mathbf{B}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{d}_{\mathscr{A}}}\right]^T \mathbf{a} = \frac{\partial J_h^{\boldsymbol{\alpha}}}{\partial \mathbf{d}_{\mathscr{A}}}^T. \tag{5.30}$$

Vector $\mathbf{b}$ then follows from substituting $\mathbf{a}$ in (5.29). Furthermore, we have

$$\left[\frac{\mathrm{d}\mathbf{c}_{\mathscr{A}}}{\mathrm{d}\boldsymbol{\alpha}}\right]^T \mathbf{b} = \left[\left[\frac{\mathrm{d}\mathbf{c}_{\mathscr{I}}}{\mathrm{d}\boldsymbol{\alpha}}\right]^T, \left[\frac{\partial \mathbf{c}_{\mathscr{B}}}{\partial \boldsymbol{\alpha}}\right]^T\right] \begin{bmatrix}\mathbf{b}_{\mathscr{I}} \\ \mathbf{b}_{\mathscr{B}}\end{bmatrix} = \left[\frac{\partial \mathbf{c}_{\mathscr{B}}}{\partial \boldsymbol{\alpha}}\right]^T \mathbf{q}, \tag{5.31}$$

where

$$\mathbf{q} = -\left[\frac{\partial \mathbf{F}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathscr{B}}}\right]^T \mathbf{e} + \mathbf{b}_{\mathscr{B}} \quad \text{with} \quad \mathbf{e} = \left[\frac{\partial \mathbf{F}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathscr{I}}}\right]^{-T} \mathbf{b}_{\mathscr{I}}. \tag{5.32}$$

We compute the matrix-vector product

$$\mathbf{e} = \left[\frac{\partial \mathbf{F}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathscr{I}}}\right]^{-T} \mathbf{b}_{\mathscr{I}} \quad \text{from the solution of} \quad \left[\frac{\partial \mathbf{F}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathscr{I}}}\right]^T \mathbf{e} = \mathbf{b}_{\mathscr{I}}. \tag{5.33}$$

Finally, it should be noted that the matrix $[\partial_{\boldsymbol{\alpha}} \mathbf{c}_{\mathscr{B}}]$ in (5.31) depends on the collocation operator

$$\pi^{\boldsymbol{\alpha}} : \boldsymbol{V} \setminus \boldsymbol{V}^0 \to \boldsymbol{V}_h^{\boldsymbol{\alpha},\mathscr{B}}, \quad \text{with} \quad \pi^{\boldsymbol{\alpha}}(\mathbf{x}_D^{\boldsymbol{\alpha}}) = \mathbf{x}_{D,h}^{\boldsymbol{\alpha}} \tag{5.34}$$

and typically involves a sparse matrix - matrix inverse product. Upon transposing, the order of multiplication is reversed and the transposed inverse moves to the front. Therefore, we can treat matrix-vector products of the form $[\partial_{\boldsymbol{\alpha}} \mathbf{c}_{\mathscr{B}}]^T \mathbf{k}$ by inverting a sparse linear system and subsequent multiplication by a sparse matrix. We will present tangiable examples of this step in Section 5.6.
In the following, we recapitulate all the necessary steps for computing the tuple $\left(J_h^{\boldsymbol{\alpha}}, \mathrm{d}_{\boldsymbol{\alpha}} J_h^{\boldsymbol{\alpha}}\right)$ for given $\boldsymbol{\alpha}$.

S.1: Choose an appropriate spline space tuple $(\mathscr{U}_h^{\boldsymbol{\alpha}}, \boldsymbol{V}_h^{\boldsymbol{\alpha}})$.

S.2: Compute $\mathbf{x}_{D,h}^{\boldsymbol{\alpha}}$ from $\mathbf{x}_D^{\boldsymbol{\alpha}}$ using $\pi^{\boldsymbol{\alpha}}$.

S.3: Solve the nonlinear root-finding problem $\mathbf{F}_h^{\boldsymbol{\alpha}}(\mathbf{c}_{\mathscr{I}}) = \mathbf{0}$, yielding the analysis-suitable mapping $\mathbf{x}_h^{\boldsymbol{\alpha}}$.

S.4: Solve the root-finding problem $\mathbf{B}_h^{\boldsymbol{\alpha}} = \mathbf{0}$ using a suitable numerical algorithm. This yields the state variable $u_h^{\boldsymbol{\alpha}}$.

S.5: Substitute $(\mathbf{x}_h^{\boldsymbol{\alpha}}, u_h^{\boldsymbol{\alpha}}, \boldsymbol{\alpha})$ in $J(\cdot,\cdot,\cdot)$ to compute $J_h^{\boldsymbol{\alpha}}$.

S.6: Compute $\mathbf{a} \to \mathbf{b} \to \mathbf{e} \to \mathbf{q}$ and finally $\mathrm{d}_{\boldsymbol{\alpha}} J_h^{\boldsymbol{\alpha}}$ using (5.28) to (5.33).

Due to the approximate nature of the tuple $(u_h^{\boldsymbol{\alpha}}, \mathbf{x}_h^{\boldsymbol{\alpha}})$, we allow for a small amount of slack in the assessment of numerical feasibility, i.e., we replace

$$\begin{aligned} g_i(u^{\boldsymbol{\alpha}}, \mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \geq 0 \quad &\longrightarrow \quad g_i(u_h^{\boldsymbol{\alpha}}, \mathbf{x}_h^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \geq \mu \quad \forall i \in \{1, \dots, N_{\neq}\} \\ h_j(u^{\boldsymbol{\alpha}}, \mathbf{x}^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) = 0 \quad &\longrightarrow \quad -\mu \leq h_j(u_h^{\boldsymbol{\alpha}}, \mathbf{x}_h^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}) \leq \mu \quad \forall j \in \{1, \dots, N_{=}\}, \end{aligned} \tag{5.35}$$

with $\mu > 0$ in (5.3). The procedure that carries out **S.1** to **S.6**, along with the relaxed constraints, is passed to a gradient-based optimization routine (such as IPOPT).

5.5.3. GRADIENT ASSEMBLY COSTS

In the following, we analyse the computational costs of assembling the gradient. The majority of the costs result from assembling sparse matrices, as well as solving sparse linear systems, such as in (5.30). In order to compute $\mathbf{x}_h^{\boldsymbol{\alpha}}$, we simultaneously assemble the quantities

$$\mathbf{F}_h^{\boldsymbol{\alpha}}(\mathbf{c}_{\mathscr{I}}^i) \quad \text{and} \quad \left[\frac{\partial \mathbf{F}_h^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathscr{I}}^i}\right] \tag{5.36}$$

during a joint element loop at the beginning of the i-th Newton iteration (see Section 5.4). As such, this routine automatically yields the matrix $\left[\partial_{\mathbf{c}_{\mathscr{I}}} \mathbf{F}_h^{\boldsymbol{\alpha}}\right]$ at the last step. In the following, we assume that $\partial\Omega_D^{\boldsymbol{\alpha}} = \emptyset$ for convenience, i.e., $u_h^{\boldsymbol{\alpha}}$ is free of Dirichlet data or the data is enforced with Nitsche's method [HH02]. If $\mathbf{B}_h^{\boldsymbol{\alpha}}$ is linear, assembling $\left[\partial_{\mathbf{d}_{\mathscr{A}}} \mathbf{B}_h^{\boldsymbol{\alpha}}\right]$ is a precursor to computing $u_h^{\boldsymbol{\alpha}}$ and hence available. If $\mathbf{B}_h^{\boldsymbol{\alpha}}$ is nonlinear, we recommend basing an iterative algorithm on Newton's method and computing the residual and its derivative in tandem, as in (5.36). As such, additional cost factors are assembling $\left[\partial_{\mathbf{c}_{\mathscr{A}}} \mathbf{B}_h^{\boldsymbol{\alpha}}\right]$ and solving a number of sparse linear equations. Due to the nonlinear nature of $\mathbf{F}_h^{\boldsymbol{\alpha}}$, the cost of computing $\mathrm{d}_{\boldsymbol{\alpha}} J_h^{\boldsymbol{\alpha}}$ is of the same order as a discrete evaluation of $J(\cdot,\cdot,\cdot)$ (regardless of the length of $\boldsymbol{\alpha}$).

Finally, we note that the discrete constraint gradients associated with the g_i and h_j are efficiently computed by replacing $J_h^{\boldsymbol{\alpha}}$ by the corresponding term in (5.23) and repeating steps (5.28) to (5.33). Hereby the required matrices can be reused from the assembly of the gradient.

5.5.4. MEMORY-SAVING STRATEGIES IN LARGE-SCALE APPLICATIONS

In light of enabling large-scale optimization as well as the prospect of extending the presented methodology to volumetric applications, in the following, we discuss ways to avoid the memory-consuming assembly of the matrices involved in computing $u_h^{\boldsymbol{\alpha}}$, $\mathbf{x}_h^{\boldsymbol{\alpha}}$ and $J_h^{\boldsymbol{\alpha}}$ and their derivatives.

Memory-saving strategies are based on the observation that matrices only appear in the

form of matrix-vector products during the assembly of $\mathrm{d}_{\boldsymbol{\alpha}} J_h^{\boldsymbol{\alpha}}$. Let $\mathbf{B} = \mathbf{B}(\ldots, \mathbf{q}, \ldots)$. Then, we have

$$\left[\frac{\partial \mathbf{B}(\ldots, \mathbf{q}, \ldots)}{\partial \mathbf{q}}\right] \mathbf{a} \simeq \frac{B(\ldots, \mathbf{q} + \epsilon \mathbf{a}, \ldots) - B(\ldots, \mathbf{q}, \ldots)}{\epsilon}, \tag{5.37}$$

for $\epsilon > 0$ small. As such, in steps (5.28) to (5.33), matrix-vector products can be approximated using (5.37). Since Krylov-subspace (KS) methods such as GMRES [SS86] only require matrix-vector products, we combine a KS-method with (5.37) for solving linear systems as they appear in, e.g., equation (5.30). Reminiscent of *Newton-Krylov* [KK04], this principle may be extended to the computation of $u_h^{\boldsymbol{\alpha}}$ and $\mathbf{x}_h^{\boldsymbol{\alpha}}$, hence completely bypassing matrix assembly in steps **S.1** to **S.6**. Hereby, we regard the cumulative error contribution to $\mathrm{d}_{\boldsymbol{\alpha}} J_h^{\boldsymbol{\alpha}} \simeq \mathrm{d}_{\boldsymbol{\alpha}} J^{\boldsymbol{\alpha}}$ as negligible compared to other sources (such as the model error and the truncation resulting from the numerical scheme). The optimal choice of ϵ is discussed in [KK04].

5.6. EXAMPLES

In this section we apply the methodology from Section 5.5.1 to selected test cases. We consider the first example a validation test case, in which the exact minimizer can be computed exactly (up to machine precision). Hereby, we compare the results of VBA (see Section 5.5) to the exact minimum. Furthermore, we compare the VBA results to those resulting from taking the tuple $(\mathscr{U}_h^{\boldsymbol{\alpha}}, \boldsymbol{V}_h^{\boldsymbol{\alpha}})$ static (SBA). In the second case, we consider the design of a cooling element, whereby the plausibility of the outcome can only be assessed using physical reasoning.

Both examples have been carefully selected in order to be geometrically challenging. We implemented the scheme from Section 5.5 in the open-source Python library *Nutils* [vZvZV+19].

5.6.1. A VALIDATION EXAMPLE WITH KNOWN EXACT SOLUTION

We are considering the example of a domain fenced-off by four parametric curves that are given by an envelope function multiplied by a cosine, whereby the amplitude of the cosine is a degree of freedom in $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$. We define the function

$$d(s) = \underbrace{\left(\frac{g(s) - g(0)}{g(0.5) - g(0)}\right)}_{\text{envelope function}} \underbrace{\left(\frac{1 - \cos(\omega \pi s)}{2}\right)}_{\text{trigonometric component}}, \quad \text{where} \quad g(s) = \exp\left(-\frac{\left(s - \frac{1}{2}\right)^2}{2\sigma^2}\right), \tag{5.38}$$

with $(\omega, \sigma) = (6, 0.2)$. Note that $d(0) = d(1) = 0$ and $d(0.5) = 1$, $d(s)$ and the envelope function are depicted in Figure 5.1. Let $\partial\hat{\Omega} = \bar{\gamma}_S \cup \bar{\gamma}_E \cup \bar{\gamma}_N \cup \bar{\gamma}_W$, where the γ_β, $\beta \in \{S, E, N, W\}$ denote the southern, eastern, northern, and western boundary of $\partial\hat{\Omega}$, respectively. The contour parameterization $\mathbf{x}_D^{\boldsymbol{\alpha}} : \partial\hat{\Omega} \to \partial\Omega^{\boldsymbol{\alpha}}$ reads:

$$\mathbf{x}_D^{\boldsymbol{\alpha}}(\boldsymbol{\xi}) = \begin{cases} (\xi_1, \alpha_1 d(\xi_1))^T & \boldsymbol{\xi} \in \bar{\gamma}_S \\ (1 - \alpha_2 d(\xi_2), \xi_2)^T & \boldsymbol{\xi} \in \bar{\gamma}_E \\ (\xi_1, 1 - \alpha_3 d(\xi_1))^T & \boldsymbol{\xi} \in \bar{\gamma}_N \\ (\alpha_4 d(\xi_2), \xi_2)^T & \boldsymbol{\xi} \in \bar{\gamma}_W \end{cases}, \tag{5.39}$$

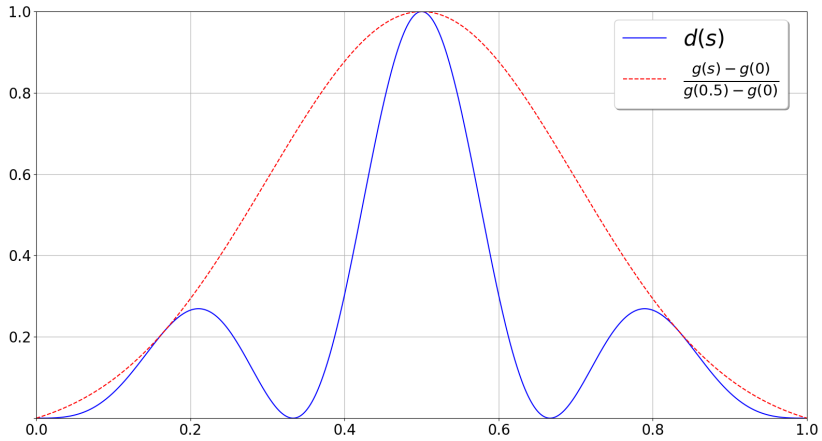


Figure 5.1: Plot of $d(s)$ (see (5.38)) and the corresponding envelope function over the interval $s \in [0,1]$.

while

$$\boldsymbol{\lambda} = \{\boldsymbol{\alpha} \in \mathbb{R}^4 \mid \mathbf{0} \leq \boldsymbol{\alpha} \leq \tfrac{2}{5}\mathbb{1}\}, \tag{5.40}$$

where $\mathbb{1}$ is a vector of ones.
Here, we base $\pi^{\boldsymbol{\alpha}} : \mathcal{V}^2 \setminus \boldsymbol{V}^{\mathbf{0}} \to \boldsymbol{V}_h^{\boldsymbol{\alpha},\mathscr{B}}$ (see Section 5.5) on an $L_2(\partial\hat{\Omega})$ projection. For given $\boldsymbol{V}_h^{\boldsymbol{\alpha},\mathscr{B}}$, we hence have

$$\left[\frac{\partial \mathbf{c}_{\mathscr{B}}}{\partial \boldsymbol{\alpha}}\right]^T = -\left[\frac{\partial \mathbf{D}^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\alpha}}\right]^T \left[\frac{\partial \mathbf{D}^{\boldsymbol{\alpha}}}{\partial \mathbf{c}_{\mathscr{B}}}\right]^{-T}, \quad \text{where} \quad \mathbf{D}_i^{\boldsymbol{\alpha}} = \int_{\partial\hat{\Omega}} \boldsymbol{\sigma}_i^{\mathscr{B}} \cdot \left(\mathbf{x}_D^{\boldsymbol{\alpha}} - \mathbf{x}_{D,h}^{\boldsymbol{\alpha}}(\mathbf{c}_{\mathscr{B}})\right) \mathrm{d}\gamma \tag{5.41}$$

and $\boldsymbol{\sigma}_i^{\mathscr{B}} \in \left[\boldsymbol{V}_h^{\boldsymbol{\alpha},\mathscr{B}}\right]$. In (5.41), we take matrix-vector products in the same way as in Section 5.5. We base our state variable residual on the following PDE problem:

$$-\Delta_{\mathbf{x}^{\boldsymbol{\alpha}}} u^{\boldsymbol{\alpha}} = -\Delta_{\mathbf{x}^{\boldsymbol{\alpha}}} f^{\boldsymbol{\alpha}} \quad \text{in } \hat{\Omega}, \quad \text{s.t.} \quad u^{\boldsymbol{\alpha}}\big|_{\partial\hat{\Omega}} = f^{\boldsymbol{\alpha}}, \quad \text{where} \quad f^{\boldsymbol{\alpha}} = \det\left[\frac{\partial \mathbf{x}^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\xi}}\right] \tag{5.42}$$

and $\Delta_{\mathbf{x}^{\boldsymbol{\alpha}}}$ denotes the Laplace-Beltrami operator corresponding to $\mathbf{x}^{\boldsymbol{\alpha}}$. Clearly, the exact solution of (5.42) satisfies $u^{\boldsymbol{\alpha}} = f^{\boldsymbol{\alpha}}$. We derive the weak form of (5.42) and implement the boundary conditions using Nitsche's method. This leads to

$$\left(\mathbf{B}_h^{\boldsymbol{\alpha}}\right)_i = \left(\nabla(u_h^{\boldsymbol{\alpha}} - f^{\boldsymbol{\alpha}}), \nabla\phi_i\right)_{\Omega_h^{\boldsymbol{\alpha}}} - \int_{\partial\Omega_h^{\boldsymbol{\alpha}}} \phi_i \frac{\partial u_h^{\boldsymbol{\alpha}}}{\partial \mathbf{n}} \mathrm{d}\gamma - \int_{\partial\Omega_h^{\boldsymbol{\alpha}}} (u_h^{\boldsymbol{\alpha}} - f^{\boldsymbol{\alpha}}) \frac{\partial \phi_i}{\partial \mathbf{n}} \mathrm{d}\gamma + \eta_i \int_{\partial\Omega_h^{\boldsymbol{\alpha}}} (u_h^{\boldsymbol{\alpha}} - f^{\boldsymbol{\alpha}}) \phi_i \mathrm{d}\gamma, \tag{5.43}$$

with $\phi_i \in \left[\mathscr{U}_h^{\boldsymbol{\alpha}}\right]$. Here, $\partial/\partial\mathbf{n}$ denotes the outward normal derivative with respect to $\Omega_h^{\boldsymbol{\alpha}}$ and $\eta_i \gg 1$ is a penalty parameter. We use

$$\eta_i = \begin{cases} cI_i^{-1} & I_i > 0 \\ 0 & \text{else} \end{cases}, \quad \text{where} \quad I_i = \int_{\partial\Omega_h^{\boldsymbol{\alpha}}} \phi_i \mathrm{d}\gamma \quad \text{and} \quad c = 10^3. \tag{5.44}$$

The objective function reads:

$$J^{\boldsymbol{\alpha}} = \int_{\hat{\Omega}} u^{\boldsymbol{\alpha}} \mathrm{d}S + \frac{1}{2} \|\boldsymbol{\alpha}\|^2 \implies J_h^{\boldsymbol{\alpha}} = \int_{\hat{\Omega}} u_h^{\boldsymbol{\alpha}} \mathrm{d}S + \frac{1}{2} \|\boldsymbol{\alpha}\|^2 \tag{5.45}$$

and there are no further constraints. Since $u^{\boldsymbol{\alpha}} = \partial_{\boldsymbol{\xi}} \mathbf{x}^{\boldsymbol{\alpha}}$, we have

$$\int_{\hat{\Omega}} u^{\boldsymbol{\alpha}} \mathrm{d}S = \mathrm{Area}(\Omega^{\boldsymbol{\alpha}}) = 1 - \sum_i \alpha_i A, \quad \text{where} \quad A = \int_{[0,1]} d(s) \mathrm{d}s. \tag{5.46}$$

We compute the exact value of A up to machine precision, which yields $A \simeq 0.2374$. The exact minimum over $\boldsymbol{\alpha} \in \boldsymbol{\lambda}$ is assumed at $\boldsymbol{\alpha}* = A\mathbb{1}$ and yields $J^{\boldsymbol{\alpha}*} \simeq 0.8873$. The contours of the resulting domain $\Omega^{\boldsymbol{\alpha}*}$ are depicted in Figure 5.2.

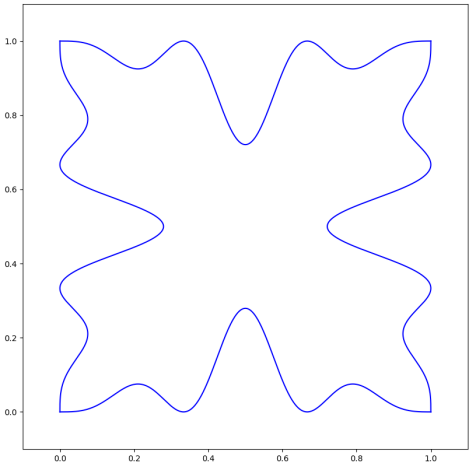


Figure 5.2: The contours of the domain $\Omega^{\boldsymbol{\alpha}*}$ that correspond to the exact minimizer $\boldsymbol{\alpha}*$.

For increasingly fine $(\mathscr{U}_h^{\boldsymbol{\alpha}}, \boldsymbol{V}_h^{\boldsymbol{\alpha}})$, the minimum of the discretized optimization problem should converge to the exact minimum, allowing us to test the consistency of the scheme. In the following, we discuss how to choose the tuple $(\mathscr{U}_h^{\boldsymbol{\alpha}}, \boldsymbol{V}_h^{\boldsymbol{\alpha}})$ during each iteration. We start by dividing $\hat{\Omega}$ into a structured set of elements, resulting from the bivariate knot vector $\Xi^{p_1,p_2} = \Xi^{p_1} \times \mathscr{H}^{p_2}$, where the p_i denote the polynomial degree. Here, we restrict ourselves to bicubic bases, i.e., $p_1 = p_2 = 3$, with maximum regularity. With points A to C (see Section 5.5) in mind, we repeatedly refine the

$$\phi_i \in \left[\mathscr{V}_h^{\boldsymbol{\alpha},\mathscr{B}}\right], \quad \text{where, as before,} \quad \mathscr{V}_h^{\boldsymbol{\alpha}} \times \mathscr{V}_h^{\boldsymbol{\alpha}} = \boldsymbol{V}_h^{\boldsymbol{\alpha}},$$

until the Dirichlet data is approximated sufficiently well.
Here, $\mathscr{V}_h^{\boldsymbol{\alpha}}$ is initialized to the coarse-grid basis resulting from Ξ^{p_1,p_2}. Let the i-th contribution to the projection residual be denoted by $r_i(\mathbf{x}_{D,h}^{\boldsymbol{\alpha}})$, where

$$R(\mathbf{x}_{D,h}^{\boldsymbol{\alpha}})^2 = \frac{1}{2} \int_{\partial\hat{\Omega}} \left\| \mathbf{x}_D^{\boldsymbol{\alpha}} - \mathbf{x}_{D,h}^{\boldsymbol{\alpha}} \right\|^2 \mathrm{d}\gamma = \frac{1}{2} \sum_i \int_{\partial\hat{\Omega}} \phi_i \left\| \mathbf{x}_D^{\boldsymbol{\alpha}} - \mathbf{x}_{D,h}^{\boldsymbol{\alpha}} \right\|^2 \mathrm{d}\gamma \equiv \frac{1}{2} \sum_i r_i^2(\mathbf{x}_{D,h}^{\boldsymbol{\alpha}}). \tag{5.47}$$

Here, we made use of the fact that the ϕ_i form a partition of unity on $\partial\hat{\Omega}$.
We refine $\phi_i \in \mathcal{V}_h^{\boldsymbol{\alpha},\mathscr{B}}$ whenever $r_i(\mathbf{x}_{D,h}^{\boldsymbol{\alpha}})$ exceeds a threshold μ_i. The threshold is of the form

$$\mu_i = \frac{\mu}{\sqrt{\|\phi_i\|_{L_2(\partial\hat{\Omega})}}}, \tag{5.48}$$

where μ is a small positive constant that tunes the accuracy of $\mathbf{x}_{D,h}^{\boldsymbol{\alpha}}$.
As a next step, we compute $\mathbf{x}_h^{\boldsymbol{\alpha}}$ using the methodology from Section 5.4. As the choice of $\boldsymbol{V}_h^{\boldsymbol{\alpha}}$, at this point, is solely based on accurately resolving the boundary contours, it may be too optimistic (in terms of the number of inner DOFs) for computing a folding-free mapping. In the case of folding, we apply a posteriori refinement to *defective* elements, i.e., elements $\mathscr{E} \subset \hat{\Omega}$ on which

$$\det\left[\frac{\partial \mathbf{x}_h^{\boldsymbol{\alpha}}}{\partial \boldsymbol{\xi}}\right](\mathbf{p}) < 0$$

for some $\mathbf{p} \in \mathscr{E}$, by refining all $\phi_i \in \left[\mathcal{V}_h^{\boldsymbol{\alpha}}\right]$ that are non-vanishing on $\mathscr{E}$. The defective mapping is prolonged to the refined space and serves as an initial guess for recomputing it from the enriched space. This step may be repeated until $\boldsymbol{V}_h^{\boldsymbol{\alpha}}$ is such that $\mathbf{x}_h^{\boldsymbol{\alpha}} \in \boldsymbol{V}_h^{\boldsymbol{\alpha}}$ is folding-free. Note that, although the proposed methodology is robust in practice, it may lead to over-refinement. The methodology may be combined with the refinement strategies proposed in [HAM20] (see Chapter 4), which avoid over-refinement.

Remark. Here, we base the selection of $\boldsymbol{V}_h^{\boldsymbol{\alpha}}$ on a posteriori strategies, which necessitates recomputing $\mathbf{x}_h^{\boldsymbol{\alpha}}$ after each refinement. Choosing the coarse-grid basis properly (i.e., not too coarse), we typically did not encounter more than $1-2$ a posteriori refinements in the cases considered in this work. Fortunately, the defective mappings can be used as an initial guess for the recomputed one, significantly reducing computational costs.
Reliable a priori refinement strategies are however desirable and constitute a topic for future research.

After achieving bijectivity, additional refinement can be performed in order to further improve the quality of the mapping. A posteriori strategies that rely on the *Winslow* functional [CI97] are discussed in [HAM20].
Upon completion, we are in the possession of an analysis-suitable $\mathbf{x}_h^{\boldsymbol{\alpha}} : \hat{\Omega} \to \Omega_h^{\boldsymbol{\alpha}}$ from the appropriately refined $\boldsymbol{V}_h^{\boldsymbol{\alpha}}$. As a next step, we choose a suitable space $\mathscr{U}_h^{\boldsymbol{\alpha}}$. Heuristically, there exists a strong correlation between the regions in which $\mathcal{V}_h^{\boldsymbol{\alpha}}$ has been refined in order to yield an analysis-suitable $\mathbf{x}_h^{\boldsymbol{\alpha}}$ and the regions that ought to be refined in order to accurately approximate $u^{\boldsymbol{\alpha}}$. As such, we initialize $\mathscr{U}_h^{\boldsymbol{\alpha}}$ to the current choice of $\mathcal{V}_h^{\boldsymbol{\alpha}}$. In this chapter, we always base $\mathscr{U}_h^{\boldsymbol{\alpha}}$ on $\mathcal{V}_h^{\boldsymbol{\alpha}}$ or a (possibly repeated) uniform h-refinement thereof. However, for more flexibility, we briefly recapitulate possible feature-based refinement strategies.
In all cases, plausible a priori (aPr) strategies refine elements that are too large (on $\Omega_h^{\boldsymbol{\alpha}}$), while a posterori (aPos) strategies depend on the underlying PDE problem. In the case of (5.42), aPos refinement can be based on the *strong residual norm*

$$m_{\mathrm{SR}} = \int_{\Omega_h^{\boldsymbol{\alpha}}} \left(\Delta u_h^{\boldsymbol{\alpha}} - \Delta f^{\boldsymbol{\alpha}}(\mathbf{x}_h^{\boldsymbol{\alpha}})\right)^2 \mathrm{d}S + \mathscr{F}(u_h^{\boldsymbol{\alpha}}|_{\partial\hat{\Omega}}), \tag{5.49}$$

where $\mathscr{F}(\cdot) : \mathscr{U}_h^{\boldsymbol{\alpha},\mathscr{B}} \to \mathbb{R}^+$ is a suitably-chosen penalty term that gauges how well the boundary condition is resolved by Nitsche's method. Note that (5.49) requires that the entries of $\mathbf{x}_h^{\boldsymbol{\alpha}}$ are elements from $C^2(\hat{\Omega})$, which is satisfied if we utilize bicubics with maximum regularity. Equation (5.49) may then be decomposed into the basis function wise contributions (as in (5.47)) or serve as a cost function for *dual weighted residual* (DWR) based aPos refinement [Ran04].
Alternatively, the *weak residual norm* m_{WR}, with

$$m_{\mathrm{WR}} = \sum_i c_{\mathrm{WR,i}}^2 \quad \text{and} \quad c_{\mathrm{WR,i}} = B(u_h^{\boldsymbol{\alpha}}, \mathbf{x}_h^{\alpha}, \boldsymbol{\alpha}, \psi_i) \tag{5.50}$$

may be utilized. Here, the ψ_i are taken from a space $\bar{\mathscr{U}} \supset \mathscr{U}_h^{\boldsymbol{\alpha}}$ that results from uniformly refining $\mathscr{U}_h^{\boldsymbol{\alpha}}$ in p or h. For more details, we refer to [GENN12].
Upon completion of an adequate state variable approximation $u_h^{\boldsymbol{\alpha}}$, we are in the position to assemble the tuple $\left(J_h^{\boldsymbol{\alpha}}, \mathrm{d}_{\boldsymbol{\alpha}} J_h^{\boldsymbol{\alpha}}\right)$ utilizing the principles from Section 5.5. All the required steps are summarized in Figure 5.3.

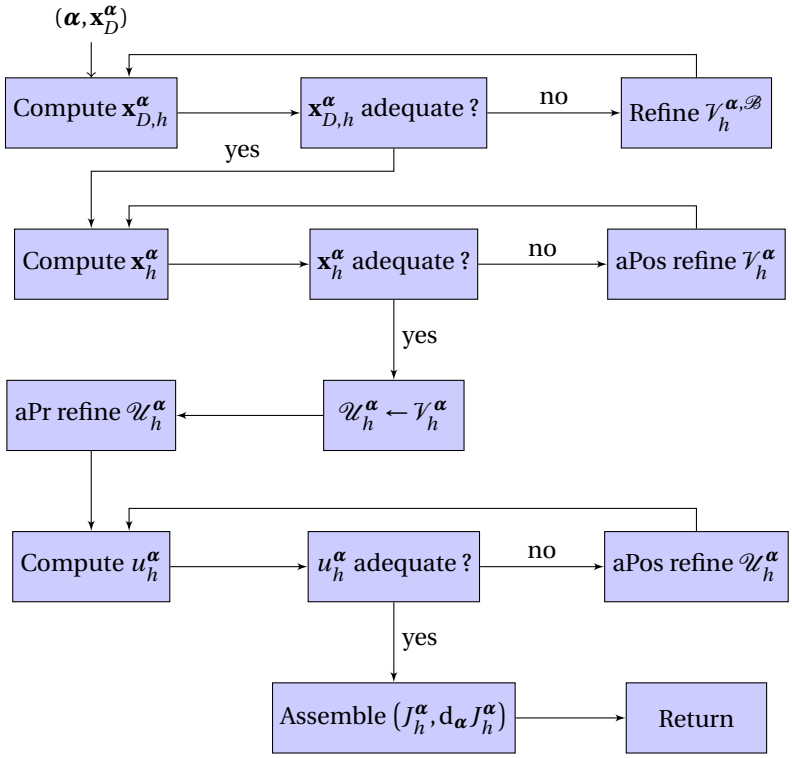


Figure 5.3: Block diagram summarizing all the steps required for computing the tuple $\left(J_h^{\boldsymbol{\alpha}}, \mathrm{d}_{\boldsymbol{\alpha}} J_h^{\boldsymbol{\alpha}}\right)$.

In the following, we present the results of a computational approach for various values of μ (see equation (5.48)) and u_{ref}, where u_{ref} refers to the number of aPr h-refinements of

$\mathscr{U}_h^{\boldsymbol{\alpha}}$ with respect to $\mathscr{V}_h^{\boldsymbol{\alpha}}$. For this, the procedure that corresponds to Figure 5.3 has been passed to a SLSQP [Kra88] routine. In all cases, we use the initial guess $\boldsymbol{\alpha}_0 = \mathbf{0}$.

μ \ u_{ref}	0	1	2
10^{-2}	8.2×10^{-3}	4.6×10^{-4}	7.7×10^{-6}
10^{-4}	3.7×10^{-3}	1.6×10^{-4}	5.5×10^{-6}
10^{-6}	2.7×10^{-4}	1.3×10^{-5}	5.2×10^{-7}

(a) Table showing $|\min J_h^{\boldsymbol{\alpha}} - \min J^{\boldsymbol{\alpha}}|$.

μ \ u_{ref}	0	1	2
10^{-2}	4	2	3
10^{-4}	4	3	3
10^{-6}	2	3	3

(b) #iterations required until convergence.

Table 5.1: Tables showing $|\min J_h^{\boldsymbol{\alpha}} - \min J^{\boldsymbol{\alpha}}|$ (a) and the required number of iterations until convergence is reached (b) for various combinations of (μ, u_{ref}).

5

μ \ u_{ref}	0	1	2
10^{-2}	483.5	389	452
10^{-4}	735.5	676	676
10^{-6}	1145	1460	1460

(a) Average #DOFs for $\mathbf{x}_h^{\boldsymbol{\alpha}}$.

μ \ u_{ref}	0	1	2
10^{-2}	241.75	625	2641
10^{-4}	367.75	1169	4337
10^{-6}	572.5	2737	10609

(b) Average #DOFs for $u_h^{\boldsymbol{\alpha}}$.

Table 5.2: Tables showing the average of the number of DOFs involved in computing $\mathbf{x}_h^{\boldsymbol{\alpha}}$ (a) and $u_h^{\boldsymbol{\alpha}}$ (b) (over all iterations) for various combinations of (μ, u_{ref}).

Table 5.1 shows the discrepancy between the exact objective function minimum and its numerical approximation $|\min J_h^{\boldsymbol{\alpha}} - \min J^{\boldsymbol{\alpha}}|$ (a) and the required number of iterations until convergence is reached (b), for all possible combinations of $\mu = (10^{-2}, 10^{-4}, 10^{-6})$ and $u_{\text{ref}} = (0, 1, 2)$. In all cases, we initialized $\left[\mathscr{V}_h^{\boldsymbol{\alpha}}\right]$ to a bicubic B-spline basis with 7 elements per coordinate direction and maximum regularity. Finally, Tables 5.3 and 5.4

n_e \ u_{ref}	0	1	2
12	8.4×10^{-3}	5.4×10^{-4}	6.5×10^{-6}
16	7.2×10^{-3}	4.1×10^{-4}	1.1×10^{-5}
23	4.7×10^{-3}	2.4×10^{-4}	8.3×10^{-6}

(a) Table showing $|\min J_h^{\boldsymbol{\alpha}} - \min J^{\boldsymbol{\alpha}}|$.

n_e \ u_{ref}	0	1	2
12	4	3	2
16	4	3	2
23	4	3	2

(b) #iterations required until convergence.

Table 5.3: Tables showing $|\min J_h^{\boldsymbol{\alpha}} - \min J^{\boldsymbol{\alpha}}|$ (a) and the required number of iterations until convergence is reached (b) in the SBA case for various combinations of (n_e, u_{ref}).

show the corresponding results from a static basis approach. Hereby, n_e denotes the number of elements we used per coordinate direction. Their values have been carefully selected to yield roughly the same number of DOFs associated with both $\mathbf{x}_h^{\boldsymbol{\alpha}}$ and $u_h^{\boldsymbol{\alpha}}$ as

the average number of DOFs in the VBA-results.
Table 5.1 (a) clearly demonstrates the consistency of the scheme, whereby discrepancies as low as $\sim 0.5 \times 10^{-6}$ are achieved. Comparing the VBA to the SBA results, tables 5.1 (a) and 5.3 (a) demonstrate that VBA outperforms SBA in terms of accuracy, where an up to $\sim$ 10-fold error reduction of VBA over SBA can be observed. The total number of iterations required until convergence is achieved is comparable for VBA and SBA and never exceeds the number of four iterations.

n_e \ u_{ref}	0	1	2
12	450	450	450
16	722	722	722
23	1352	1352	1352

(a) #DOFs for $\mathbf{x}_h^{\boldsymbol{\alpha}}$.

n_e \ u_{ref}	0	1	2
12	225	729	2601
16	361	1225	4489
23	676	2401	9025

(b) #DOFs for $u_h^{\boldsymbol{\alpha}}$.

Table 5.4: Tables showing the number of DOFs involved in computing $\mathbf{x}_h^{\boldsymbol{\alpha}}$ (a) and $u_h^{\boldsymbol{\alpha}}$ (b) in the SBA case for various combinations of (n_e, u_{ref}).

5.6.2. DESIGNING A COOLING ELEMENT

We are considering the design of a cooling element of dimension $\Delta x = 2$ and $\Delta y = 1$. In this example, there are four active coolers whose positions can slide in the direction tangential to $\partial\Omega^{\boldsymbol{\alpha}}$ and to a lesser extend in the normal direction (see Figure 5.4). Further degrees of freedom are their radii R_i. Hence, the state vector is given by $\boldsymbol{\alpha} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, R_1, R_2, R_3, R_4)$, which is comprised of 12 DOFs. The surface cooling rate for the i-th active cooler $\mathscr{C}_i$ reads:

$$h_-^i(\mathbf{x}) = \frac{1}{20} \frac{R_i^3}{\|\mathbf{x} - \mathbf{x}_i\|^2} \left(u_h^{\boldsymbol{\alpha}}(\mathbf{x}) - T_\infty \right), \tag{5.51}$$

where $T_\infty = 0$ denotes the ambient temperature. A heat source delivers a constant heat influx given by

$$N_{\text{tot}} = N_{\text{in}}^W + N_{\text{in}}^{\text{int}}, \tag{5.52}$$

where N_{in}^W denotes the influx at the western boundary, while $N_{\text{in}}^{\text{int}}$ denotes the influx delivered directly to the cooling element through an additional source term which satisfies

$$N_{\text{in}}^{\text{int}} = \int_{\Omega^{\boldsymbol{\alpha}}} A \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}_0\|^2}{2\sigma^2} \right) \mathrm{d}S, \tag{5.53}$$

where $A = \frac{N_{\text{tot}}}{4\pi\sigma^2}$, $\mathbf{x}_0 = (1.5, 0.25)^T$ and $\sigma = 0.1$. Note that changing the domain of integration from $\Omega^{\boldsymbol{\alpha}}$ to $\mathbb{R}^2$ in (5.53) yields a value of $N_{\text{in}}^{\text{int}} = N_{\text{tot}}/2$. As N_{tot} is a constant quantity, we necessarily have $N_{\text{in}}^W = N_{\text{tot}} - N_{\text{in}}^{\text{int}}$. The surface heat flux density $h_W : \Gamma_W^{\boldsymbol{\alpha}} \to \mathbb{R}$ at the

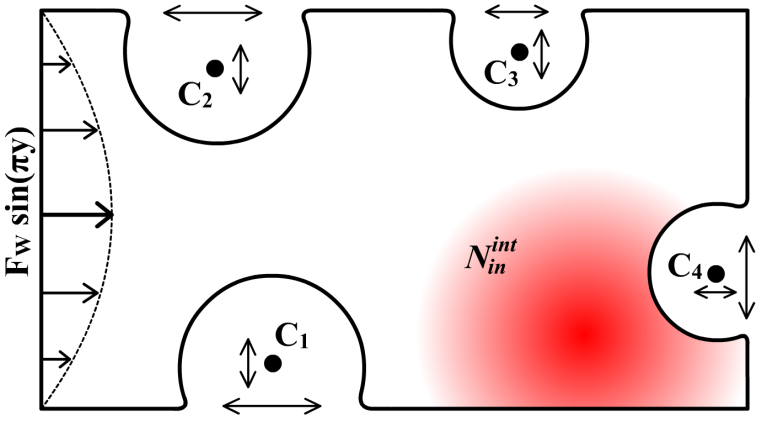


Figure 5.4: The cooling element design template. Here, the centers of the active coolers are depicted by small black dots. Their positions constitute degrees of freedom in the design space, as well as their radii.

western boundary $\Gamma_W^{\boldsymbol{\alpha}} \subset \partial\Omega^{\boldsymbol{\alpha}}$ is of the form $h_W(y) = F_W(\Omega^{\boldsymbol{\alpha}}) \sin(\pi y)$. Therefore, we have

$$N_{\text{in}}^W = N_{\text{tot}} - N_{\text{in}}^{\text{int}} = \int_{\Gamma_W^{\boldsymbol{\alpha}}} F_W \sin(\pi y) \mathrm{d}\gamma. \tag{5.54}$$

Hence

$$F_W(\Omega^{\boldsymbol{\alpha}}) = \frac{N_{\text{tot}}\pi}{2}\left(1 - \int_{\Omega^{\boldsymbol{\alpha}}} \frac{1}{4\pi\sigma^2} \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_0\|^2}{2\sigma^2}\right)\mathrm{d}S\right). \tag{5.55}$$

The relationship between $u^{\boldsymbol{\alpha}}$ and the (uniform) temperature of the heat source $T^{\boldsymbol{\alpha}}$ reads:

$$N_{\text{tot}} = A_1(\Omega^{\boldsymbol{\alpha}}) \int_{\Gamma_W^{\boldsymbol{\alpha}}} (T^{\boldsymbol{\alpha}} - u^{\boldsymbol{\alpha}}) \sin(\pi y) \mathrm{d}\gamma + A_2(\Omega^{\boldsymbol{\alpha}}) \int_{\Omega^{\boldsymbol{\alpha}}} (T^{\boldsymbol{\alpha}} - u^{\boldsymbol{\alpha}}) \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_0\|^2}{2\sigma^2}\right)\mathrm{d}S, \tag{5.56}$$

where

$$A_1(\Omega^{\boldsymbol{\alpha}}) = \frac{\pi}{2}\left(1 - \frac{W(\Omega^{\boldsymbol{\alpha}})}{4\pi\sigma^2}\right) \quad \text{and} \quad A_2(\Omega^{\boldsymbol{\alpha}}) = \frac{W(\Omega^{\boldsymbol{\alpha}})}{8\pi^2\sigma^4}, \tag{5.57}$$

with

$$W(\Omega^{\boldsymbol{\alpha}}) = \int_{\Omega^{\boldsymbol{\alpha}}} \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_0\|^2}{2\sigma^2}\right)\mathrm{d}S. \tag{5.58}$$

Inverting (5.56) gives:

$$T^{\boldsymbol{\alpha}}(u^{\boldsymbol{\alpha}}, \Omega^{\boldsymbol{\alpha}}) = \frac{N_{\text{tot}} + A_1(\Omega^{\boldsymbol{\alpha}}) \int_{\Gamma_W^{\boldsymbol{\alpha}}} u^{\boldsymbol{\alpha}} \sin(\pi y) \mathrm{d}\gamma + A_2(\Omega^{\boldsymbol{\alpha}}) \int_{\Omega^{\boldsymbol{\alpha}}} u^{\boldsymbol{\alpha}} \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_0\|^2}{2\sigma^2}\right)\mathrm{d}S}{\frac{2}{\pi} A_1(\Omega^{\boldsymbol{\alpha}}) + W(\Omega^{\boldsymbol{\alpha}}) A_2(\Omega^{\boldsymbol{\alpha}})}. \tag{5.59}$$

Remark. The rationale behind $A_1(\Omega^{\boldsymbol{\alpha}})$ and $A_2(\Omega^{\boldsymbol{\alpha}})$ in (5.59) can be understood as follows: given $N_{\text{tot}} = 1$, suppose the cooling element width were to be contracted to $\Delta x \to 0$. We have $\lim_{\Delta x \to 0} W(\Omega^{\boldsymbol{\alpha}}) = 0$. In the limit, the temperature of the heat source should be fully determined by the first term on the right hand side of (5.56), which is the case because $\lim_{\Delta x \to 0}(A_1, A_2) = (\frac{\pi}{2}, 0)$. As such, a constant influx of $1 = N_{\text{tot}} = N_{\text{in}}^W$ means

$$T^{\boldsymbol{\alpha}} - u^{\boldsymbol{\alpha}}|_{\Gamma_W^{\boldsymbol{\alpha}}} = 1.$$

Conversely, suppose $\Omega^{\boldsymbol{\alpha}}$ were to be replaced by $\mathbb{R}^2$. Then, the dependency is divided equally among both terms since for

$$W(\Omega^{\boldsymbol{\alpha}} = \mathbb{R}^2) = 2\pi\sigma^2, \quad \text{we have} \quad (A_1, A_2) = \left(\frac{\pi}{4}, \frac{1}{4\pi\sigma^2}\right).$$

So, for $T^{\boldsymbol{\alpha}} - u^{\boldsymbol{\alpha}} = 1$, both terms contribute the same factor of $\frac{1}{2}$ to the right hand side of (5.56).

The weak state equation is based on the following PDE-problem:

$$\begin{aligned} -d\Delta u^{\boldsymbol{\alpha}} &= -f u^{\boldsymbol{\alpha}} + A\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_0\|^2}{2\sigma^2}\right) \quad \text{in } \Omega^{\boldsymbol{\alpha}} \\ \text{s.t.} \quad d\left.\frac{\partial u^{\boldsymbol{\alpha}}}{\partial \mathbf{n}}\right|_{\partial\Omega^{\boldsymbol{\alpha}}} &= \begin{cases} -h_{\text{cooling}} + F_W \sin(\pi y) & \mathbf{x} \in \bar{\Gamma}_W^{\boldsymbol{\alpha}} \\ -h_{\text{cooling}} & \mathbf{x} \in \partial\Omega^{\boldsymbol{\alpha}} \setminus \bar{\Gamma}_W^{\boldsymbol{\alpha}} \end{cases}, \end{aligned} \tag{5.60}$$

where

$$h_{\text{cooling}} = \sum_{i=1}^{4} h_-^i \quad \text{and} \quad f = 10^{-3} \quad \text{denotes the internal dissipation rate.} \tag{5.61}$$

The i-th entry of the discretized weak state equation residual reads:

$$\begin{aligned} \left(\mathbf{B}_h^{\boldsymbol{\alpha}}\right)_i &= d\left(\nabla u_h^{\boldsymbol{\alpha}}, \nabla\phi_i\right)_{\Omega_h^{\boldsymbol{\alpha}}} + \int_{\Omega_h^{\boldsymbol{\alpha}}} f u^{\boldsymbol{\alpha}} \phi_i \mathrm{d}S \\ &+ \int_{\Omega_h^{\boldsymbol{\alpha}}} A\exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_0\|^2}{2\sigma^2}\right)\left(\frac{\pi}{2}\overline{\phi}_i - \phi_i\right)\mathrm{d}S + \sum_{j=1}^{4}\int_{\partial\Omega_h^{\boldsymbol{\alpha}}} \phi_i h_-^j \mathrm{d}\gamma - \frac{\pi}{2} N_{\text{tot}}\overline{\phi}_i, \end{aligned} \tag{5.62}$$

with A as in (5.53), $d = 0.8$,

$$\overline{\phi}_i = \int_{\Gamma_W^{\boldsymbol{\alpha}}} \phi_i \sin(\pi y)\mathrm{d}\gamma \quad \text{and} \quad \phi_i \in \left[\mathscr{U}_h^{\boldsymbol{\alpha}}\right]. \tag{5.63}$$

We are minimizing the manufacturing costs of the cooling element such that the heat source temperature does not exceed the value of $T_{\max} = 80$. The problem reads:

$$\begin{aligned} J\left(u^{\boldsymbol{\alpha}}, \Omega^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}\right) &\to \min_{\boldsymbol{\alpha}} \\ \text{s.t.} \quad T_{\max} - T^{\boldsymbol{\alpha}} &\geq 0 \\ \boldsymbol{\alpha} &\in \boldsymbol{\lambda}, \end{aligned} \tag{5.64}$$

where

$$J\left(u^{\boldsymbol{\alpha}}, \Omega^{\boldsymbol{\alpha}}, \boldsymbol{\alpha}\right) = \int_{\Omega^{\boldsymbol{\alpha}}} 1 \mathrm{d}S + \sum_{i=1}^{4} C_{\mathrm{CE}} R_i^2, \quad \text{with} \quad C_{\mathrm{CE}} = \frac{100}{\pi}. \tag{5.65}$$

Furthermore, the feasible design space $\boldsymbol{\lambda}$ is the space of all $\boldsymbol{\alpha}$ such that the active coolers do not overlap and the genus of $\Omega^{\boldsymbol{\alpha}}$ does not change (allowing for shape optimization without topology changes). This leads to a total of 30 (partly nonlinear) inequalities.
A major challenge is deciding where to place the active coolers and what radii to use. Increasing the radius means additional cooling but also additional manufacturing costs and decreased cooling element area, decreasing the heat capacity and the channel heat conductivity. Furthermore, placing a cooler close to the internal heat source (see (5.53)) reduces the amount of internal influx, increasing the influx amplitude F_W (see (5.54)) at $\Gamma_W^{\boldsymbol{\alpha}}$ for compensation.
We are considering the case $N_{\mathrm{tot}} = 10$ and follow the same approach as in Section 5.6.1 with $\mu = 0.5 \times 10^{-3}$ and $u_{\mathrm{ref}} = 1$. Since $\mathbf{x}^{\boldsymbol{\alpha}}$ is a continuous function of the input state vector, we improve the efficiency by storing the tuples $(\boldsymbol{\alpha}^i, \mathbf{c}_{\mathscr{A}}^i, \boldsymbol{V}_h^{\boldsymbol{\alpha},i})$ (see Section 5.5) after each iteration. Whenever some $\boldsymbol{\alpha}^i$ with $\|\boldsymbol{\alpha} - \boldsymbol{\alpha}^i\| < \epsilon$ is found in the database, the corresponding mapping $\mathbf{x}_h^{\boldsymbol{\alpha},i} \in \boldsymbol{V}_h^{\boldsymbol{\alpha},i}$ is prolonged to the coarsest element segmentation of $\hat{\Omega}$ that is compatible with both $\boldsymbol{V}_h^{\boldsymbol{\alpha},i}$ and the current $\boldsymbol{V}_h^{\boldsymbol{\alpha}}$. Upon completion, it is restricted to $\boldsymbol{V}_h^{\boldsymbol{\alpha}}$, which yields the vector $\mathbf{c}_{\mathscr{A}}^R = \left(\mathbf{c}_{\mathscr{B}}^R, \mathbf{c}_{\mathscr{I}}^R\right)^T$. The weights corresponding to the inner DOFs, $\mathbf{c}_{\mathscr{I}}^R$, are extracted and then used as an initial guess for the root-finding problem (5.13). We have noticed this to lead to a tremendous speedup, in particular during the last iterations, in which $\boldsymbol{\alpha}$ varies only slightly. Hereby, the required number of iterations is reduced from typically four to as few as one.

Remark. This principle may be extended to higher than zeroth-order database interpolation.

Here, we use $\epsilon = 0.05$. A feasible initial guess is created by picking one of the coolers and increasing its radius until $T^{\boldsymbol{\alpha}} < T_{\max}$. The initial design is depicted in Figure 5.5. As in section 5.6.1, the routine that computes $J_h^{\boldsymbol{\alpha}}$, $\mathrm{d}_{\boldsymbol{\alpha}} J_h^{\boldsymbol{\alpha}}$ and the constraints is passed to an SLSQP optimizer.
Figures 5.6 (a) to 5.6 (d) show the cooling element after 4, 7, 10 and 13 iterations. Convergence is reached after 15 iterations and the corresponding design is depicted in Figure 5.7. The final design reduces the manufacturing costs from the initial $J_h^{\boldsymbol{\alpha}} = 10.66$ to $J_h^{\boldsymbol{\alpha}} = 6.29$.
A striking difference between the initial and all intermediate designs is the improved heat conductivity within the channel, leading to a more homogeneous temperature (and one that is higher on average). This is not surprising. As the cooling efficiency is linear in the difference between the temperature at the boundaries and the ambient temperature $T_\infty = 0$, a higher average temperature implies higher average cooling efficiency.
The final design places a modestly-sized cooler $\mathscr{C}_1$ at the center of the southern boundary and a similarly-sized cooler $\mathscr{C}_2$ at the western part of the northern boundary. To its right, a slightly larger cooler $\mathscr{C}_3$ is placed while a small cooler $\mathscr{C}_4$ is placed at eastern boundary close to $y = 0.4$.

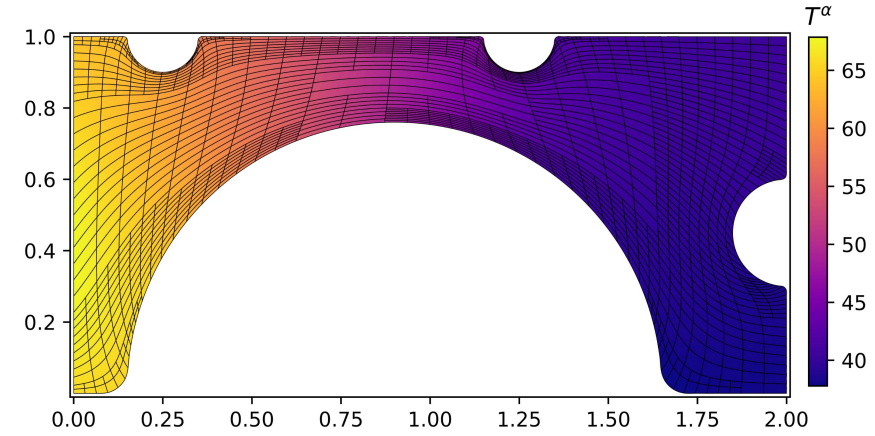


Figure 5.5: The initial guess passed to the minimization routine.

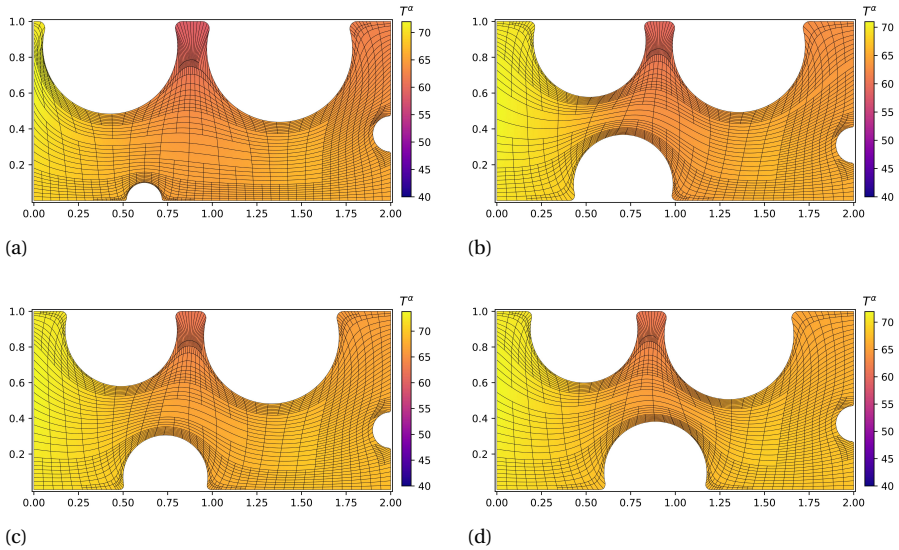


Figure 5.6: The cooling element after 4 (a), 7 (b), 10 (c) and 13 (d) iterations.

Compared to the initial design, one big cooler has been replaced by several modestly-sized ones, improving the channel heat conductivity and by that the cooling cost efficiency. The slightly larger size of $\mathscr{C}_3$ compared to $\mathscr{C}_2$ can be explained by the internal heat source centered at $\mathbf{x}_0 = (1.5, 0.25)^T$. The small radius of $\mathscr{C}_4$ may be explained by the fact that increasing its size reduces the amount of internal influx area, leading to a larger influx at $\Gamma_W^{\boldsymbol{\alpha}}$ instead. As such, we regard the final design as plausible, adding more credibility to the proposed numerical scheme.

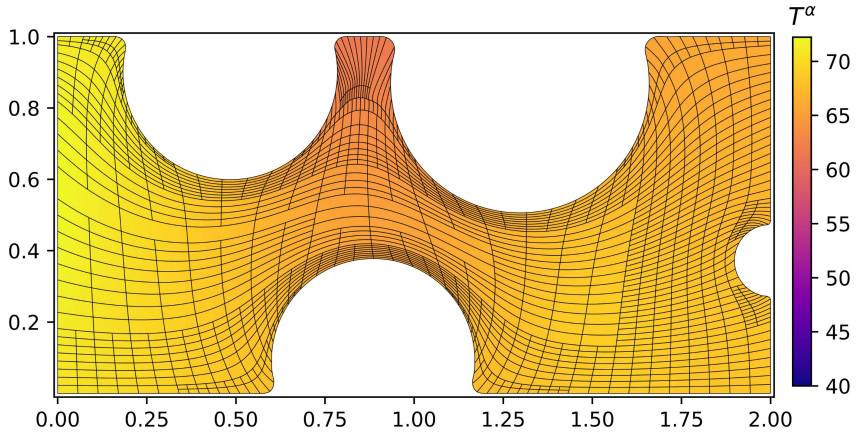


Figure 5.7: The final cooling element design after 15 iterations.

5.7. CHAPTER CONCLUSIONS

In this chapter, we proposed an IGA-based shape optimization algorithm in which the parameterization is included in the problem formulation in the form of an additional PDE-constraint. This has enabled us to derive a fully symbolical expression for the gradient of the objective function, allowing for gradient-based optimization. The discretization of the equations has been accomplished with the so-called *variable basis approach* (VBA) in which a new THB-spline basis is chosen during each iteration based on the current requirements, such as accurately resolving the geometry contours and particular features of the state equation solution. This leads to a highly flexible scheme in which folding due to numerical truncation is automatically repaired through THB-enabled local refinement.

We have tested the scheme by applying it to two examples. In the first example, we compared the numerical solution to the known exact solution and concluded that the scheme is consistent. Comparing the VBA-approach to an approach in which the basis is taken static (SBA) furthermore revealed that VBA-enabled feature-based refinement leads to a $\sim$ 10-fold error reduction over SBA at a comparable total number of DOFs. This discrepancy may be further increased by employing more proficient a priori and a posteriori refinement techniques. In the second example, we considered the design of a cooling element. Unlike in the first example, the exact minimizer was unknown, however, the optimization routine converged to a design that we consider plausible. In both cases, the scheme succeeded in fully automatically parameterizing a wide range of geometries which would be too complex for other symbolically-differentiable parameterization strategies (such as Coon's Patch) at the expense of leading to a nonlinear problem. Finally, we briefly discussed possible memory-saving strategies for large-scale optimization and (possible) future implementations of the scheme with support for volumetric applications. Furthermore, the scheme is straightforwardly enhanced to support multipatch parameterizations by adopting the mixed FEM EGG algorithm introduced in [HMV19] (see Chapter 3).

REFERENCES

[Aza09] Boris Nikolaevich Azarenok. Generation of structured difference grids in two-dimensional nonconvex domains using mappings. *Computational Mathematics and Mathematical Physics*, 49(5):797–809, 2009.

[BH08] Y Bazilevs and TJR Hughes. Nurbs-based isogeometric analysis for the computation of flows about rotating components. *Computational Mechanics*, 43(1):143–150, 2008.

[Bot10] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[BZ09] Lorenz T Biegler and Victor M Zavala. Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575–582, 2009.

[CI97] AA Charakhch'yan and SA Ivanenko. A variational form of the winslow grid generator. *Journal of Computational Physics*, 136(2):385–398, 1997.

[CRBH06] J Austin Cottrell, Alessandro Reali, Yuri Bazilevs, and Thomas JR Hughes. Isogeometric analysis of structural vibrations. *Computer methods in applied mechanics and engineering*, 195(41-43):5257–5296, 2006.

[FH99] Gerald Farin and Dianne Hansford. Discrete coons patches. *Computer Aided Geometric Design*, 16(7):691–700, 1999.

[FŠJ15] Antonella Falini, Jaka Špeh, and Bert Jüttler. Planar domain parameterization with THB-splines. *Computer Aided Geometric Design*, 35:95–108, 2015.

[GENN12] Jens Gravesen, Anton Evgrafov, Dang-Manh Nguyen, and Peter Nørtoft. Planar parametrization in isogeometric analysis. In *International Conference on Mathematical Methods for Curves and Surfaces*, pages 189–212. Springer, 2012.

[GJS12] Carlotta Giannelli, Bert Jüttler, and Hendrik Speleers. THB-splines: The truncated basis for hierarchical splines. *Computer Aided Geometric Design*, 29(7):485–498, 2012.

[HAM20] Jochen Hinz, Michael Abdelmalik, and Matthias Möller. Goal-oriented adaptive THB-spline schemes for pde-based planar parameterization, 2020.

[HCB05] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *CMAME*, 194:4135–4195, 2005.

[HH02] Anita Hansbo and Peter Hansbo. An unfitted finite element method, based on nitsche's method, for elliptic interface problems. *Computer methods in applied mechanics and engineering*, 191(47-48):5537–5552, 2002.

[HJMV20] Jochen Hinz, Andrzej Jaeschke, Matthias Möller, and Cornelis Vuik. The role of pde-based parameterization techniques in gradient-based IGA shape optimization applications. *arXiv preprint arXiv:2001.10921*, 2020.

[HMV18] J Hinz, M Möller, and C Vuik. Elliptic grid generation techniques in the framework of isogeometric analysis applications. *Computer Aided Geometric Design*, 2018.

[HMV19] Jochen Hinz, Matthias Möller, and Cornelis Vuik. An IGA framework for pde-based planar parameterization with arbitrary interface continuity. *arXiv preprint arXiv:1904.03009*, 2019.

[Hol84] John H Holland. Genetic algorithms and adaptation. In *Adaptive Control of Ill-Defined Systems*, pages 317–333. Springer, 1984.

[Jae15] A. Jaeschke. Isogeomeric analysis for compressible flows with application in turbomachinery. Master's thesis, TU Delft, 2015.

[KK04] Dana A Knoll and David E Keyes. Jacobian-free newton–krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[Kne26] Hellmuth Kneser. Losung der aufgabe 41. *Jahresber. Deutsche Meth.*, pages 123–124, 1926.

[KOŁ14] Zbigniew Kacprzyk and Katarzyna Ostapska-Łuczkowska. Isogeometric analysis as a new fem formulation-simple problems of steady state thermal analysis. *Procedia Engineering*, 91:87–92, 2014.

[KP12] Steven G Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2012.

[Kra88] Dieter Kraft. A software package for sequential quadratic programming. *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.

[MEGG11] Nguyen Dang Manh, Anton Evgrafov, Allan Roulund Gersborg, and Jens Gravesen. Isogeometric shape optimization of vibrating membranes. *Computer Methods in Applied Mechanics and Engineering*, 200(13-16):1343–1353, 2011.

[NG13] Peter Nørtoft and Jens Gravesen. Isogeometric shape optimization in fluid mechanics. *Structural and Multidisciplinary Optimization*, 48(5):909–925, 2013.

[NJ10] Thien Nguyen and Bert Jüttler. Parameterization of contractible domains using sequences of harmonic maps. In *International conference on curves and surfaces*, pages 501–514. Springer, 2010.

[Rad26] Tibor Radó. Aufgabe 41. *Jahresber. Deutsch. Math.-Verein*, 35:49, 1926.

[Ran04] Rolf Rannacher. Adaptive finite element methods in flow computations. *Recent Advances in Adaptive Computation. Contemporary Mathematics*, 383:183–176, 2004.

[SS86] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[vZvZV+19] Gertjan van Zwieten, Joost van Zwieten, Clemens Verhoosel, Eivind Fonn, Timo van Opstal, and Wijnand Hoitinga. Nutils, June 2019.

[WB06] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[WFC08] Wolfgang A Wall, Moritz A Frenzel, and Christian Cyron. Isogeometric structural shape optimization. *Computer methods in applied mechanics and engineering*, 197(33-40):2976–2988, 2008.

[XMDG10] Gang Xu, Bernard Mourrain, Régis Duvigneau, and André Galligo. Optimal analysis-aware parameterization of computational domain in isogeometric analysis. In *International Conference on Geometric Modeling and Processing*, pages 236–254. Springer, 2010.

[XMDG11] Gang Xu, Bernard Mourrain, Régis Duvigneau, and André Galligo. Parameterization of computational domain in isogeometric analysis: methods and comparison. *Computer Methods in Applied Mechanics and Engineering*, 200(23-24):2021–2031, 2011.

[YHC13] Minho Yoon, Seung-Hyun Ha, and Seonho Cho. Isogeometric shape design optimization of heat conduction problems. *International Journal of Heat and Mass Transfer*, 62:272–285, 2013.

6

Spline-Based Parameterization Techniques for Twin-Screw Machine Geometries

This chapter is based on the publication from [HMV18b] and constitutes the first large-scale application of the techniques presented in Chapter 2. In particular, it improves and automates the methodology from Section 2.10.3 and introduces a framework capable of generating parameterizations for the interior of counter-rotating rotary-screw compressor geometries at every rotational angle θ.
The proposed framework is augmented with the techniques from Chapter 3 and applied to co-rotating twin-screw extruders in Chapter 7.

The fully automated generation of computational meshes for twin-screw machine geometries constitutes a mandatory aspect for their numerical simulation but proves to be a challenging task in practice. Therefore, the successful generation of computational meshes requires sophisticated mathematical tools. Commercially available classical mesh generators can produce high quality meshes from no more than a description of the rotor contours. However, since we are particularly interested in numerical simulations using the principles of *Isogeometric Analysis* (IGA), a spline-based geometry description rather than a classical mesh is needed.
In this chapter, we propose a practical approach for the automated generation of spline-based twin-screw machine geometry parameterizations in two spatial dimensions. For this purpose, we adopt the principles of *Elliptic Grid Generation* and present a parameterization algorithm that is compatible with an automated simulation pipeline based on the principles of IGA.

To demonstrate the proposed techniques, we apply them to an example geometry and present the resulting parameterizations. Furthermore, we generate a volumetric parameterization by stacking a large number of planar cross section in the z-direction. Finally, we give a qualitative explanation of how the discussed techniques can be utilized to perform shape optimization on a variable rotor-pitch.

6.1. INTRODUCTION

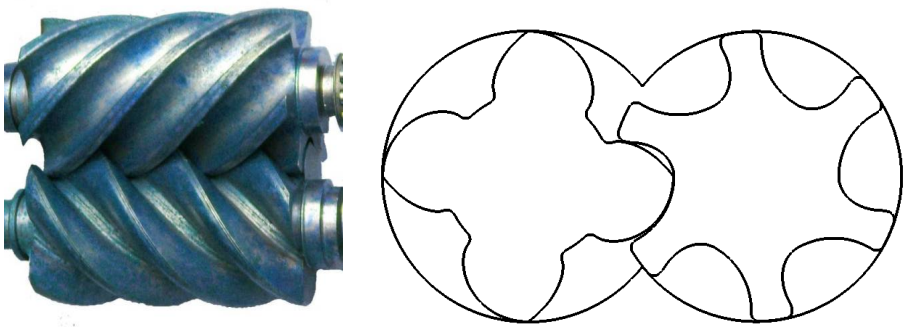


Figure 6.1: Rotary-screw compressor (Wikipedia, file: *Lysholm_screw_rotors.jpg*) (left) and a cross section showing the rotor profiles with casing (right).

The generation of analysis-suitable meshes for twin-screw geometries constitutes the first step towards the numerical simulation and shape optimization of twin-screw machines. However, the full automation of this process remains difficult, often contributing substantially to the software development and computational costs. On the one hand, this is caused by the inherent difficulty of generating a mesh from no more than a description of the boundary contours, and, on the other hand, further aggravated by the challenging characteristics of twin-screw geometries, such as tiny clearances and rapidly changing gap sizes (see Figure 6.1). To the best of our knowledge, there exist two commercial structured mesh generators that only require a description of the geometry contours as input: *twin-mesh* [TM] and *SCORG* [SCO], both being capable of exporting directly into *ANSYS CFX* format [ANS]. We are particularly interested in using Isogeometric Analysis (IGA) [HCB05] techniques to perform shape optimization on twin-screw machine geometries with variable rotor pitch. Even though the commercial mesh generators produce high-quality classical meshes, they cannot be used in an IGA-setting where a spline-based parameterization of the target geometry is mandatory. This is the main motivation for the techniques presented in this chapter.
The general idea of spline-based parameterization techniques is finding a continuous mapping operator $\mathbf{x}:\hat{\Omega}\rightarrow\Omega$, comprised of higher-order spline functions, that maps the entire parametric domain $\hat{\Omega}$ onto the target geometry Ω (or an approximation thereof). The spline-based parameterization $\mathbf{x}$ is then directly used for IGA-based simulation. Unlike in the classical case, application-specific features such as boundary layers are typically added after the geometry description has been completed. This is accomplished by performing knot refinement on $\mathbf{x}$ (see section 6.2) and has a negligible impact on the

overall computational costs.
A volumetric parameterization is accomplished by generating a large number of planar parameterizations at various discrete rotational angles θ_i and stacking them in the z-direction. For such an approach to make sense, it is required that the parameterization pipeline yields mappings whose components are continuous functions of the rotational angle θ. Hereby, higher-order continuity furthermore tends to improve parameterization quality. This translates to the requirement of smoothly varying control points of the mapping $\mathbf{x}$ as a function of the rotation angle θ. Therefore, it is furthermore desirable to, if possible, avoid topology changes in the planar slices, even though this may be a challenging task. Variable pitches are conveniently accomplished by a tighter or wider stacking of the discrete slices.
It should be mentioned that a spline-based description can be turned back into a classical mesh by performing a large number of function evaluations in $\mathbf{x}$ and connecting the resulting point cloud by linear edges. Hereby, the evaluation points determine the properties of the mesh and have to be chosen wisely. However, this topic will not be covered in this chapter.
For the purpose of generating folding-free planar parameterizations using spline functions, we adopt the principles of *Elliptic Grid Generation* (EGG) and present a numerical scheme that is suitable for an IGA-based computational approach. In Section 6.2, we briefly review the concept of (B-)spline basis functions while in Section 6.3, we present the various possible topologies along with their advantages and disadvantages. In section 6.4, we discuss the principles of EGG and in sections 6.7 and 6.8, we develop the computational approach along with the parameterization strategy we employ for the test cases presented in section 6.9

6.2. B-SPLINES

B-splines are piecewise-polynomial functions that can be constructed so as to satisfy various continuity properties at the places where the polynomial segments connect. Their properties are determined by the entries of the so-called *knot vector*

$$\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}. \tag{6.1}$$

The knot vector is a non-decreasing sequence of parametric values $\xi_i \subset [0,1]$ that determine the boundaries of the segments on which the spline basis is polynomial. Selecting some polynomial order p, the p-th order spline-functions $N_{i,p}$ are constructed recursively, utilizing the relation (with $\frac{0}{0} \equiv 0$)

$$N_{i,q}(\xi) = \frac{\xi - \xi_i}{\xi_{i+1} - \xi_i} N_{i,q-1}(\xi) + \frac{\xi_{i+q+1} - \xi}{\xi_{i+q+1} - \xi_{i+1}} N_{i+1,q-1}(\xi), \tag{6.2}$$

starting from

$$N_{i,0} = \begin{cases} 1 & \text{if } \xi_i \leq \xi < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases}, \tag{6.3}$$

and iterating until $q = p$. The support of basis function $N_{i,p}$ is given by the interval $\mathscr{I}_{i,p} = [\xi_i, \xi_{i+p+1}]$ and the amount of continuous derivatives across knot ξ_j is given by

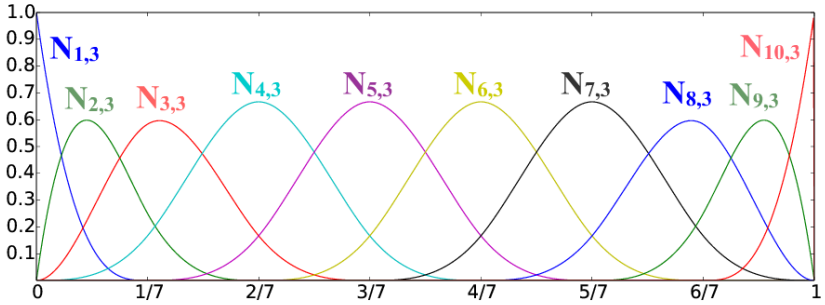


Figure 6.2: The univariate B-spline basis resulting from the knot vector Ξ from (6.5).

$p - m_j$, where m_j is the multiplicity of ξ_j in $\mathscr{I}_{i,p}$. In practice, $\xi_1 = 0$ is repeated $p+1$ times as well as ξ_{n+p+1} such that $\xi_1 = \ldots = \xi_{p+1} = 0$ and $\xi_{n+1} = \ldots = \xi_{n+p+1} = 1$. As a result, the basis $\sigma = \{N_{1,p}, \ldots, N_{n,p}\}$ forms a non-negative partition of unity on the entire parametric domain $[0,1]$, that is:

$$\sum_{i=1}^{n} N_{i,p}(\xi) = 1, \quad \text{with} \quad N_{i,p}(\xi) \geq 0 \tag{6.4}$$

for all spline functions $N_{i,p}$ [HCB05]. Figure 6.2 shows the $p = 3$ B-spline basis resulting from the knot vector

$$\Xi = \left\{0,0,0,0,\tfrac{1}{7},\tfrac{2}{7},\tfrac{3}{7},\tfrac{4}{7},\tfrac{5}{7},\tfrac{6}{7},1,1,1,1\right\}. \tag{6.5}$$

The extension to bivariate spline bases is now straight-forward: given two univariate bases $\sigma_\Xi = \{N_1, \ldots, N_n\}$ and $\sigma_\mathscr{H} = \{M_1, \ldots M_m\}$, we build a bivariate spline space $\mathcal{V}_h$ with corresponding basis $[\mathcal{V}_h] = \{w_{i,j} \mid (i,j) \in \{1,\ldots,n\} \times \{1,\ldots,m\}\}$, by means of a tensor-product, where

$$w_{i,j}(\xi,\eta) = N_i(\xi) M_j(\eta). \tag{6.6}$$

The knots corresponding to the bases σ_Ξ and $\sigma_\mathscr{H}$, with knot vectors Ξ and $\mathscr{H}$ (without knot repetitions), respectively, hereby become the boundaries of the polynomial segments, which can be regarded as the counterparts of classical elements.
We construct the mapping of a B-spline surface as follows:

$$\mathbf{x} = \sum_i \sum_j \mathbf{c}_{i,j} w_{i,j}, \tag{6.7}$$

where the $\mathbf{c}_{i,j} \in \mathbb{R}^2$ are referred to as the *control points*. We refer to the $\mathbf{c}_{i,j}$ with $i \in \{1,n\}$ or $j \in \{1,m\}$ as the *boundary control points*, while the remaining $\mathbf{c}_{i,j}$ are called *inner control points*. As $\mathbf{x}$ is a linear combination of the $w_{i,j} \in [\mathcal{V}_h]$, it inherits the local continuity properties of the basis. This implies that many geometrical features can be better captured by a clever choice of the knot multiplicities in Ξ and $\mathscr{H}$.

An additional appealing feature of spline basis functions is the possibility of knot refinement. Let

$$f = \sum_i a_i N_{i,p} \tag{6.8}$$

be a function from the linear span of the spline basis $\sigma_\Xi = \{N_{1,p}, \dots, N_{n,p}\}$. We can refine σ_Ξ by adding additional knots to Ξ, resulting in the refined basis $\sigma_{\bar{\Xi}}$. It can be shown that $\operatorname{span}\sigma_\Xi \subset \operatorname{span}\sigma_{\bar{\Xi}}$. For an algorithm to prolong the a_i to the refined basis, see [HCB05]. This principle is straightforwardly generalized to bivariate tensor-product bases.
In the following, we drop tensorial indices and introduce a global index with $(i, j) \to i + (j-1)m$. The mapping then simplifies to

$$\mathbf{x} = \sum_i \mathbf{c}_i w_i. \tag{6.9}$$

For many geometries, a parameterization with only one mapping operator is not possible, which is why several mappings that jointly parameterize the geometry have to be employed. The individual geometry segments that result from each of the mappings are referred to as *patches*.

6.3. CHOICE OF TOPOLOGY

As discussed in section 6.1, we would like to parameterize the geometry from figure 6.1 (right) for all rotational angles θ. The discrete angles θ_i then either correspond to the screw-machine at time-instances t_i in the planar case or to some cross-section of the screw-machine in the z-direction. A volumetric parameterization can therefore be acquired by parameterizing the geometry for a large number of θ_i and interpolating the planar cross-sections in the z-direction.

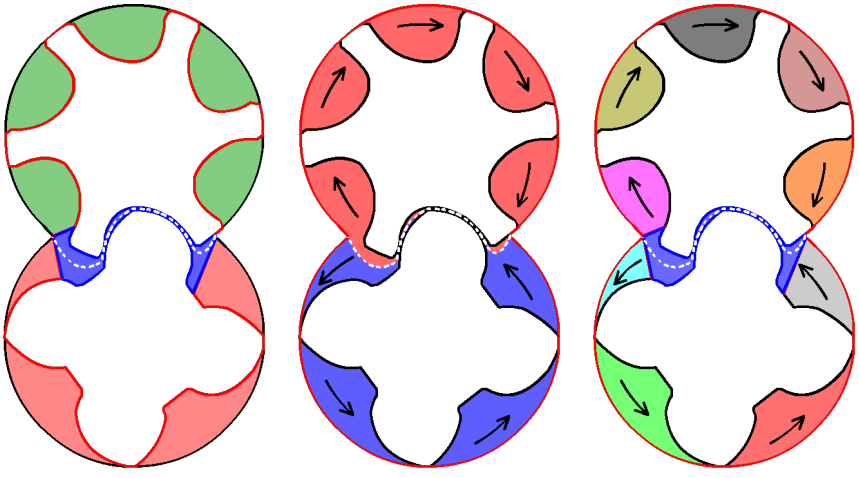


Figure 6.3: The various possible topologies for the parameterization of the fluid-part of the screw-machines.

Figure 6.3 shows the possible topologies that come to mind. Here, black lines indicate boundaries at which the grid is held fixed and red lines indicate boundaries that slide along the grid. The various patches are highlighted in different colors. Since the target geometry is of genus two (it has two *holes*), at least two patches are needed. In the following, we refer to the two points at which both casings meet as the *CUSP*-points.
Even though Figure 6.3 (right) takes advantage of the symmetries of the geometry, topology changes are unavoidable. For instance, since the blue patch bounded by the region surrounding the CUSP-points and the two rotor lobes (henceforth referred to as the *separator*) is static while the others rotate, patches (such as the grey patch) will eventually disappear, making a computational simulation with IGA-techniques difficult.
In Figure 6.3 (center), two O-type patches are employed which leads to a sliding interface (in the separator region). Sliding interfaces are numerically difficult to handle since element conformity is difficult to achieve. In an IGA-setting, the CUSP-points themselves pose an additional problem: as the patches slide along, the CUSP-point C^0-continuities can only be captured by adding repeated knots to the knot vectors. If several cross sections are interpolated in the z-direction for a volumetric parameterization, they all have to be prolonged to a unified knot vector, soon leading to an infeasibly-dense trivariate knot vector.
Therefore, we are aiming for the topology of figure 6.3 (left) in which the patches are held fixed at the casings while the rotors slide along and the separator is parameterized with one static patch. Here, no topology changes are required and no sliding interface exists. In all cases, we need to generate the dotted white curve connecting the two CUSP-points, in order to parameterize the separator using two patches with mutual element conformity as well as conformity to the C-type patches.

6.4. ELLIPTIC GRID GENERATION

Having discussed the aimed-for topology in section 6.3, we need a tool to generate analysis-suitable (i.e., bijective or folding-free) parameterizations at every discrete rotational angle θ_i, given no more than a boundary correspondence $\partial\hat{\Omega} \to \partial\Omega(\theta_i)$.
For this, we base our approach on the class of *Elliptic Grid Generation* (EGG) methods. EGG is a proficient approach in settings where computationally inexpensive algebraic methods such as [Coo67], [GH73] and [GT82] fail due to the complexity of the target geometry. However, due to the higher chance of success, higher computational costs can be expected.
Assuming Ω is topologically equivalent to the unit quadrilateral $\hat{\Omega} = [0,1] \times [0,1] \subset \mathbb{R}^2$, a mathematical operator $\mathbf{x} : \hat{\Omega} \to \Omega$ that maps $\partial\hat{\Omega}$ onto $\partial\Omega$, and is furthermore folding-free, exists and can be constructed. For this purpose, EGG imposes the Laplace-equation on the components of the inverse mapping $\mathbf{x}^{-1}$. Assuming the free topological variables are given by the tuple $\boldsymbol{\xi} = (\xi, \eta)$, the equation takes the form:

$$\Delta\mathbf{x}^{-1} = \mathbf{0} \quad \text{in} \quad \Omega \quad \text{s.t.} \quad \mathbf{x}^{-1}|_{\partial\Omega} = \partial\hat{\Omega}. \tag{6.10}$$

This system of equations is inverted and scaled in order to yield an equation for $\mathbf{x}$ that is suitable for a computational approach. The resulting equations read [TSW98]:

$$\mathscr{L}(\mathbf{x}) = \mathbf{0} \quad \text{in} \quad \hat{\Omega} \quad \text{s.t.} \quad \mathbf{x}|_{\partial\hat{\Omega}} = \partial\Omega, \tag{6.11}$$

where

$$\mathscr{L}(\mathbf{x}) = g_{22}\mathbf{x}_{\xi\xi} - 2g_{12}\mathbf{x}_{\xi\eta} + g_{11}\mathbf{x}_{\eta\eta}, \tag{6.12}$$

with $g_{11}(\mathbf{x}) = \|\mathbf{x}_\xi\|^2$, $g_{12}(\mathbf{x}) = \mathbf{x}_\xi \cdot \mathbf{x}_\eta$ and $g_{22}(\mathbf{x}) = \|\mathbf{x}_\eta\|^2$. Since the target space of $\mathbf{x}^{-1}$ is always convex, the bijectivity of the exact solution of (6.11) follows from the maximum principle [Cas91]. This justifies seeking a sufficiently accurate approximation $\mathbf{x}_h$ of $\mathbf{x}$ for the purpose of generating an analysis-suitable mapping. This property distinguishes EGG from most other meshing techniques, which may tend to produce folded meshes, due to them being less grounded in mathematical theory.

6.4.1. DISCRETIZATION

Traditionally, (6.11) is approximately solved using a finite difference approach [TSW98]. However, since this only yields a finite collection of grid points that can serve as the vertices of a classical mesh, we have to look for different options. We discretize the equations with FEA-techniques. First, we introduce the operator

$$\tilde{\mathscr{L}}(\mathbf{x}) = \frac{\mathscr{L}(\mathbf{x})}{g_{11} + g_{22} + \epsilon}, \quad \text{where} \quad \epsilon = 10^{-4}. \tag{6.13}$$

As a next step, we select a $p \geq 2$ bivariate B-spline basis $[\mathcal{V}_h] = \{w_1, \ldots, w_N\}$ with global $C^1(\hat{\Omega})$-continuity (i.e., the $w_i \in [\mathcal{V}_h]$ possess at least one continuous derivative in $\hat{\Omega}$). By $\mathcal{V}_h^\circ$, we denote the subset of $\mathcal{V}_h$ consisting of functions that vanish on $\partial\hat{\Omega}$ (corresponding to the inner control points from Section 6.2). We discretize (6.11) as follows:

$$\text{find} \quad \mathbf{x}_h \in \mathcal{V}_h^2 \quad \text{s.t.} \quad \int_{\hat{\Omega}} \boldsymbol{\sigma}_h \cdot \tilde{\mathscr{L}}(\mathbf{x}_h) \mathrm{d}\boldsymbol{\xi} = 0, \quad \forall \boldsymbol{\sigma}_h \in (\mathcal{V}_h^\circ)^2 \quad \text{and} \quad \mathbf{x}_h|_{\partial\hat{\Omega}} = \partial\Omega_h, \tag{6.14}$$

where $\partial\Omega_h$ is some approximation of $\partial\Omega$ compatible with $(\mathcal{V}_h \setminus \mathcal{V}^\circ)^2$.
Let $\mathscr{I} = \{1, \ldots, N\}$ and let $\mathscr{I}_I$ be the index-set corresponding to $\mathcal{V}_h^\circ \subset \mathcal{V}_h$. Furthermore, let $\mathscr{I}_B = \mathscr{I} \setminus \mathscr{I}_I$. The mapping $\mathbf{x}_h \in \mathcal{V}_h^2$ is of the form:

$$\mathbf{x}_h = \sum_{i \in \mathscr{I}_I} \mathbf{z}_i w_i + \sum_{j \in \mathscr{I}_B} \mathbf{z}_j w_j. \tag{6.15}$$

Here, the $\mathbf{z}_i \in \mathbb{R}^2$ denote the inner control points and the $\mathbf{z}_j \in \mathbb{R}^2$ the boundary control points. The latter follow from a regression of the input point cloud and are taken such that $\mathbf{x}_h|_{\partial\hat{\Omega}} \simeq \partial\Omega$ (see Section 6.5). As the $\mathbf{z}_j$ are known, the objective is to find the $\mathbf{z}_i$ such that $\mathbf{x}_h$ satisfies (6.14). Equation (6.14) leads to a nonlinear root finding problem of the form

$$\mathbf{F}(\mathbf{c}) = \mathbf{0}, \tag{6.16}$$

where the vector $\mathbf{c} = (\ldots, \mathbf{z}_i, \ldots)^T$ is the concatenation of the unknown inner control points $\mathbf{z}_i$.
The scaling introduced in (6.13) leads to a more scalable convergence criterion $\|\mathbf{F}(\mathbf{c})\| \leq \delta$, that is, the value of δ that corresponds to a converged solution is less sensitive to the

characteristic length-scale of the geometry (and can therefore be taken approximately equal in all cases). The choice of $\mathcal{V}_h$ and the optimal selection of the inner control points $\mathbf{z}_i$ shall be the topic of section 6.5, while the computational approach for solving (6.16) will be the topic of section 6.7.

6.5. CONTOUR APPROXIMATION AND CHOICE OF BASIS

Given four input point clouds $P_\alpha = \{\mathbf{p}_\alpha^i\}_{i=1}^{I_\alpha}$, $\alpha \in \{s,e,n,w\}$ which are assigned to each side γ_α of $\partial\hat{\Omega}$, the selection of a suitable spline basis $[\mathcal{V}_h]$ and the corresponding boundary control points $\mathbf{z}_j$ (see section 6.2) constitutes a preliminary step before (6.16) is tackled computationally. For this purpose we select a coarse initial spline space $\mathcal{V}_\square$ (whose basis is the result of two coarse univariate knot vectors) and four sets of monotone increasing parametric values $\{\xi_\alpha^i\}_{i=1}^{I_\alpha}$, each starting on $\xi_\alpha^1 = 0$ and ending on $\xi_\alpha^{I_\alpha} = 1$. Let $\mathbf{m}_s(\xi) = (\xi,0)$, $\mathbf{m}_e(\xi) = (1,\xi)$, $\mathbf{m}_n(\xi) = (\xi,1)$ and $\mathbf{m}_w(\xi) = (0,\xi)$. The objective is to select the $\mathbf{z}_j$ such that $\mathbf{x}_h(\mathbf{m}_\alpha(\xi_\alpha^i)) \simeq \mathbf{p}_\alpha^i$. To this purpose, a least-squares regression is carried out that minimizes the functional

$$R(\partial\Omega,\mathbf{d}) = \frac{1}{2}\sum_{\alpha\in\{s,e,n,w\}}\sum_{i=1}^{I_\alpha}\left\|\mathbf{x}_h(\mathbf{m}_\alpha(\xi_\alpha^i)) - \mathbf{p}_\alpha^i\right\|^2 \tag{6.17}$$

over the vector of boundary control points $\mathbf{d} = \left(\ldots,\mathbf{z}_j,\ldots\right)^T$. Hereby, it is advisable to constrain the corner control points to the corners of the input point clouds in order to avoid mismatches. In practice, (6.17) can suffer from instabilities. This is usually a result of the local amount of DOFs exceeding the local amount of points. Hence, the collocation matrix $M_\alpha = \mathcal{N}_\alpha\mathcal{N}_\alpha^T$, where

$$[\mathcal{N}_\alpha]_{ij} = \phi_i^\alpha(\mathbf{m}_\alpha(\xi_\alpha^j)), \quad \text{with} \quad [\mathcal{V}_h] \ni \phi_i^\alpha\big|_{\gamma_\alpha} \neq 0, \tag{6.18}$$

may become (nearly) rank-deficient.
We add a small least-distance penalty term to (6.17) in order to improve the stability:

$$\tilde{R}(\partial\Omega,\mathbf{d}) = \frac{1}{2}\sum_{\alpha\in\{s,e,n,w\}}\left(\sum_{i=1}^{I_\alpha}\left\|\mathbf{x}_h(\mathbf{m}_\alpha(\xi_\alpha^i)) - \mathbf{p}_\alpha^i\right\|^2 + \lambda\int_{\gamma_\alpha}\left\|\frac{\partial\mathbf{x}_h}{\partial\mathbf{t}}\right\|^2 \mathrm{d}\gamma\right), \tag{6.19}$$

where $\partial/\partial\mathbf{t}$ denotes the directional derivative in tangential direction. Here, $\lambda > 0$ is a small penalty factor whose value should be chosen small enough not to noticeably alter the outcome of (6.19), while avoiding instabilities. In practice, $\lambda = 10^{-4}$ is a robust choice.
After (6.19) has been minimized, the mismatch

$$r_{\alpha,i} = \left\|\mathbf{x}_h(\mathbf{m}_\alpha(\xi_\alpha^i)) - \mathbf{p}_\alpha^i\right\| \tag{6.20}$$

serves as a local refinement criterion. Whenever $r_{\alpha,i}$ exceeds the approximation tolerance $\mu > 0$, we place an additional knot in the center of the knot span that contains ξ_α^i associated with the knot vector of side γ_α. By this, we locally increase the resolution of the basis. Here, we only allow for (locally) dyadic refinement in order to acquire knot

vectors with a predefined structure. This reduces the total number of knots when taking the union of several knot vectors.
Above steps are repeated until the convergence criterion is reached. Upon completion, we are in the possession of the coarse- and fine-grid bases $[\mathcal{V}_\square]$ and $[\mathcal{V}_h]$ with corresponding vectors of boundary control points $\mathbf{d}_\square$ and $\mathbf{d}$, respectively. The fine-grid basis then constitutes the coarsest possible basis to resolve the boundary condition

$$\mathbf{x}|_{\partial\hat{\Omega}} \simeq \partial\Omega \tag{6.21}$$

to a user-defined accuracy, which is tuned by the choice of μ.
The tuple $(\mathcal{V}_\square, \mathbf{d}_\square)$ serves as a means to build initial guesses for the computational approach that solves (6.16) (see Section 6.7).
It should be noted that the choice of $\mathcal{V}_h$ is purely based on its capabilities to properly approximate $\partial\Omega$. However, this choice may not appropriately approximate the solution of (6.11) at every point in $\hat{\Omega}$. Heuristically, folding due to insufficient accuracy is uncommon. Should it nevertheless happen, we dyadically refine $\mathcal{V}_h$ by adding knots wherever a folds occurs and prolong $\mathbf{d}$ to the refined space $(\mathcal{V}_h^R)^2$. Equation (6.16) is solved with $\mathcal{V}_h^R$ to yield a better approximation. This step is repeated until the approximation yields a bijective mapping.

6.6. CHOICE OF PARAMETRIC VALUES

The parametric values $\{\xi_\alpha^i\}_{i=1}^{I_\alpha}$ from section 6.5 have a profound influence on the parametric quality of the resulting parameterization and have to be chosen wisely. By default, the input point clouds are chord length parameterized. Defining l_i recursively by

$$l_i = l_{i-1} + \|\mathbf{p}_i - \mathbf{p}_{i-1}\|, \tag{6.22}$$

starting with $l_1 = 0$ and ending on l_{I_α}, we let

$$\hat{\xi}_\alpha^i = \frac{l_i}{l_{I_\alpha}}. \tag{6.23}$$

A chord length parameterization then corresponds to taking $\xi_\alpha^i = \hat{\xi}_\alpha^i$.
Using (6.23) ensures that the parametric velocity at the boundaries is (approximately) constant. In the presence of extreme aspect ratios (tiny gaps), however, we have found (6.23) to lead to unsatisfactory results. To ensure the quality of the resulting parameterization, we require that (approximately) the same parametric value is assumed on either side of the gaps (see figure 6.4). For this purpose, we employ the matching algorithm proposed in [HMV18a] to two opposite point clouds P_w, P_e (or P_n, P_s) in order to match pairs of points that are too close. Let $\mathcal{I}_w = \{2, \dots, I_w - 1\}$ and $\mathcal{I}_e = \{2, \dots, I_e - 1\}$. Upon completion of the matching, we are in the possession of a finite set of matched tuples

$$\mathcal{I}^m = \{(i, j) \mid \mathbf{p}_i \in P_w \text{ and } \mathbf{p}_j \in P_e \text{ have been matched}\}. \tag{6.24}$$

The tuples $(i, j) \in \mathcal{I}^m$, $i \in \mathcal{I}_w$, $j \in \mathcal{I}_e$ can be utilized to build a reparameterization function that improves the parametric properties of the mapping. We assign the parametric

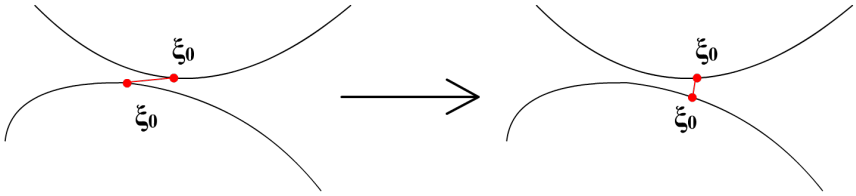


Figure 6.4: Skewed isolines due to poorly parameterized boundary contours and improved parametric properties resulting from reparameterization. Near the tiny gaps, it is of major importance that the same parametric value is assumed on both sides.

values $\{\hat{\xi}_w^i\}_{i=1}^{I_w}$ to the points $\mathbf{p}_i \in P_w$. Furthermore, we set $\xi_e^j = \hat{\xi}_w^i$ whenever $(i,j) \in \mathscr{I}^m$. The question remains what parametric value to assign to the unmatched points $\mathbf{p}_j \in P_e$. Given the set

$$\mathscr{I}^k = \left\{ j \mid \xi_e^j \text{ is known} \right\}$$

(note that $\mathscr{I}^k$ contains $\xi_e^1 = 0$ and $\xi_e^{I_e} = 1$), we carry out a monotone cubic spline-interpolation [FC80] of the values $\{\hat{\xi}_e^i\}_i,\ i \in \mathscr{I}^k$ versus the known values $\{\xi_e^i\}_i,\ i \in \mathscr{I}^k$, to yield the monotone reparameterization function $\xi_e'(\xi)$. The ξ_e^i then follow from evaluating ξ_e' in the $\hat{\xi}_e^i$, that is

$$\xi_e^i = \xi_e'\left(\hat{\xi}_e^i\right). \tag{6.25}$$

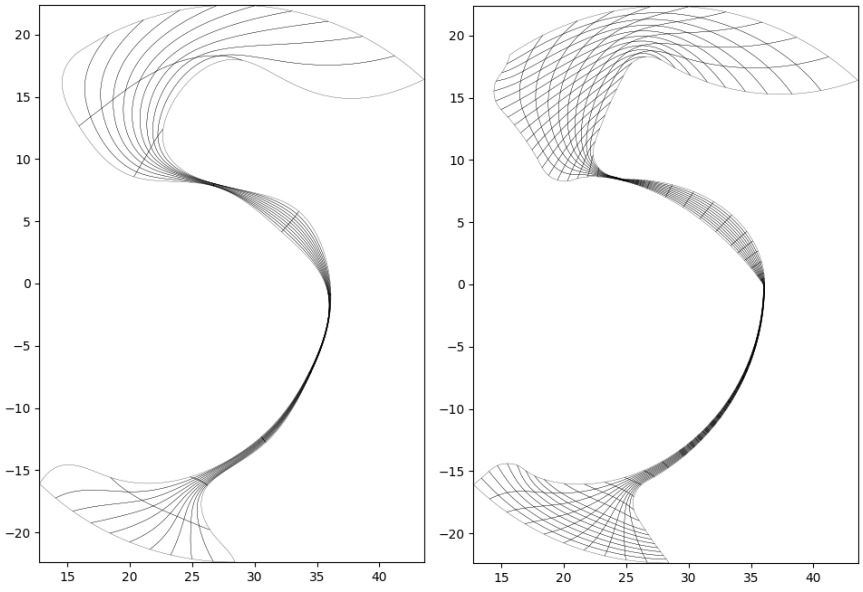
In practice, reparameterization is a crucial ingredient for the successful parameterization of the target geometry and should therefore always be employed along with the numerical approach that is the topic of Section 6.7.

6.7. COMPUTATIONAL APPROACH

After completion of the regression from section 6.5, possibly in conjunction with reparameterization (see section 6.6), we are in the position to tackle the root-finding problem from equation (6.16). The point cloud regression yields the tuples $(\mathcal{V}_\square, \mathbf{d}_\square)$ and $(\mathcal{V}_h, \mathbf{d})$ of coarse- and fine-grid spaces with corresponding boundary control points. We tackle (6.16) with a truncated Newton-approach. We first solve the coarse-grid problem $\mathbf{F}_\square(\mathbf{c}_\square) = \mathbf{0}$ using transfinite interpolation [Coo67] to generate an initial guess $\mathbf{c}_\square^0$. Both the coarse- and the fine-grid problem are of the form $\mathbf{G}(\mathbf{c}) = \mathbf{0}$, where $\mathbf{G}$ is nonlinear in $\mathbf{c}$. Given some initial guess $\mathbf{c}^0$, the new iterate is computed using the following recursive relation:

$$\left.\frac{\partial \mathbf{G}}{\partial \mathbf{c}}\right|_{\mathbf{c}=\mathbf{c}^i} \delta\mathbf{c} = -\mathbf{G}(\mathbf{c}^i) \quad \text{and} \quad \mathbf{c}^{i+1} = \mathbf{c}^i + \kappa\delta\mathbf{c}. \tag{6.26}$$

Here $0 < \kappa \leq 1$ is a truncation parameter whose optimal value is estimated from the current and updated tangents and residuals. Upon solving the coarse-grid problem $\mathbf{F}_\square(\mathbf{c}_\square) = \mathbf{0}$, the coarse-grid solution $\mathbf{c}_\square$ is prolonged to the fine-grid space $\left(\mathcal{V}_h^\circ\right)^2$ and serves as an initial guess for the fine-grid problem $\mathbf{F}(\mathbf{c}) = \mathbf{0}$.



(a) Coarse-grid solution (b) Fine-grid solution

Figure 6.5: Coarse- (a) and fine-grid (b) solution of a challenging input geometry.

In practice, the coarse-grid problem typically converges after $4-6$ iterations, while the fine-grid problem requires an additional $2-3$ iterations. Thanks to the relatively small number of DOFs associated with $\mathcal{V}_\square$, the impact of the large number of required coarse-grid iterations is manageable. Since the required number of fine-grid iterations is approximately halved, the expected speed-up is $\sim 50\%$. Furthermore, the robustness of the approach is greatly improved: thanks to the high quality of the initial guess, failure of convergence is extremely uncommon. As an example, Figures 6.5 (a) and (b) plot the parameterizations corresponding to the coarse- and fine-grid solutions of a challenging geometry. Here, we performed reparameterization (see section 6.6) on the western and eastern boundaries, keeping the western boundary chord length parameterized while letting the eastern boundary float. Convergence on the fine grid is reached within 3 nonlinear iterations.

6.8. APPLICATION TO TWIN-SCREW MACHINE GEOMETRIES

The computational approach discussed in section 6.7 constitutes the basic ingredient for the parameterization approach that will be the topic of this section. When no good initial guess is available, we employ the hierarchical approach from Section 6.7. Else, we use the initial guess to directly solve the fine-grid problem. Selecting K uniformly-spaced discrete angles θ_k from the interval $[0, \pi/2]$, the objective is to compute a planar paramterization for every θ_k. The approach consists of the following steps:

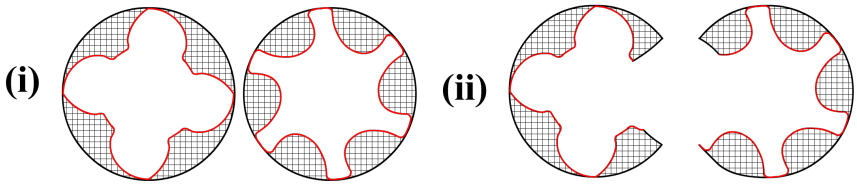


Figure 6.6: Steps (i) and (ii) of the parameterization strategy.

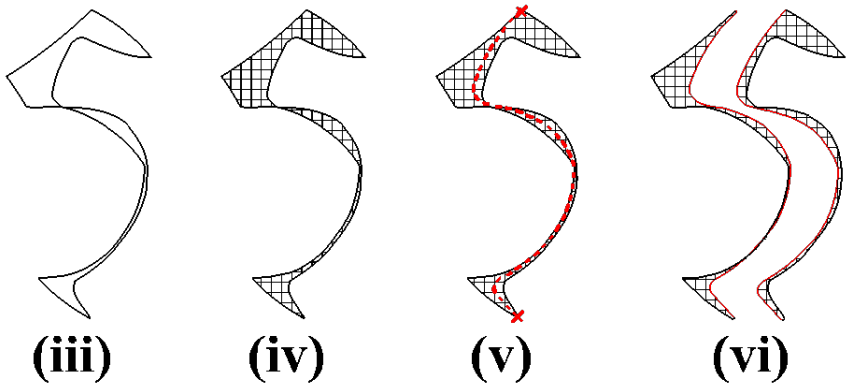


Figure 6.7: Steps (iii) to (vi) of the parameterization strategy.

1. Generate separate O-type parameterizations for the male and female rotors with casing for every θ_k.
2. Cut the O-parameterizations at the CUSP-points in order to produce two C-parameterizations for every discrete angle.
3. Combine the cuts with the male and female rotor parts to form a contour description of the separator.
4. Compute singlepatch parameterizations for the separator at every discrete angle.
5. Use the singlepatch parameterized separators to generate curves, connecting the two CUSP-points and splitting the separator in half.
6. Generate parameterizations on the left and on the right of the splitting curves using the same knot vector in η-direction to acquire a conforming interface. Use the same knot vector as for the C-grids in ξ-direction to acquire conforming separator-C-grid interfaces.

The key steps are depicted in Figures 6.6 and 6.7.
To generate the O-type parameterizations from (i), we first generate exact (chord length parameterized) cubic spline-fits through the input point clouds of the rotors and casings, using one of the FITPACK [Die95] routines. We evaluate both spline fits in N uniformly-spaced points over the parametric interval $[0,1]$ and utilize the resulting point clouds to

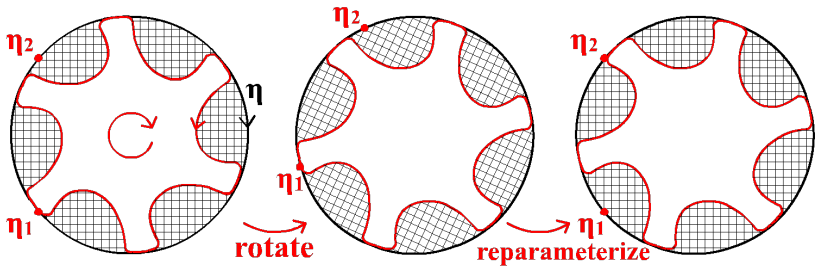


Figure 6.8: At every angle θ_k we act with the canonical rotation matrix on the control points of the spline-fit and reparameterize such that the CUSP-points always correspond to the same parametric values η_1 and η_2.

build male and female reparameterization functions η'_m and η'_f, respectively (see section 6.6). Hereby the casings are held chord length parameterized (i.e., their reparameterization functions are the identity) while we let the rotor parameterizations float.
At every discrete angle θ_k, we act with the canonical rotation matrix on the control points of the fitted spline curves and reparameterize such that the CUSP-points always coincide with the same parametric values η_1 and η_2. Since the casings are chord length parameterized, we reparameterize by a shift of $-\theta_k/(2\pi)$ in the parametric domain. The rotor spline-fits have to be shifted by the same value in the reparameterized η'-domain. Therefore, we compute the shift in η by inverting $\eta'_{m,f}$ (see figure 6.8). As a next step, we utilize the shifted spline fits to generate a large number of uniformly-spaced points which serve as an input point cloud. We utilize the reparameterization functions $\eta'_{m,f}$, which we shift in a way similar to the spline fits, to assign parametric values to the rotor point clouds.
For the O-type parameterizations, we utilize a $p \geq 2$-th order knot vector which is periodic in the η-direction while disregarding the northern and southern boundaries. We compute parameterizations corresponding to θ_k, $k = 1,\ldots,6$ utilizing the hierarchical approach from Section 6.7. Once completed, we continue filling the database by extrapolating the inner control points corresponding to $\theta_{k-6},\ldots,\theta_k$ to θ_{k+1} and utilize the result as an initial guess. Upon completion of the K slices for each rotor, we add the $p+1$-times repeated knots η_1 and η_2, which correspond to the η-values of the CUSP-points in $\hat{\Omega}$, to the knot vectors associated with the η-direction of the O-grids. This way, the two separate male and female rotor O-grids are each split into two parts: one C-grid and one grid which is discarded. By cutting in the domain, we avoid accidentally cutting the rotor lobes twice, as a result, the cuts are not (necessarily) straight.
Upon completion, our database is filled with left and right C-grid parameterizations for each discrete angle θ_k. Having completed steps (i) and (ii), a description of the separator contours can be acquired by evaluating the exact rotor spline-fits from η_1 to η_2 (on both sides) and combining the resulting point clouds with the C-grid cuts (see figure 6.9). The resulting geometries are parameterized with one patch. Instead of sequentially parameterizing all K slices, we start off by building parameterizations for each L-th slice (with $L \ll K$), keeping the western boundary chord length parameterized, while building reparameterization functions for the eastern boundary (see Section 6.6). For each slice, we employ the hierarchical approach from section 6.7. Let η'_k, denote the reparameteriza-

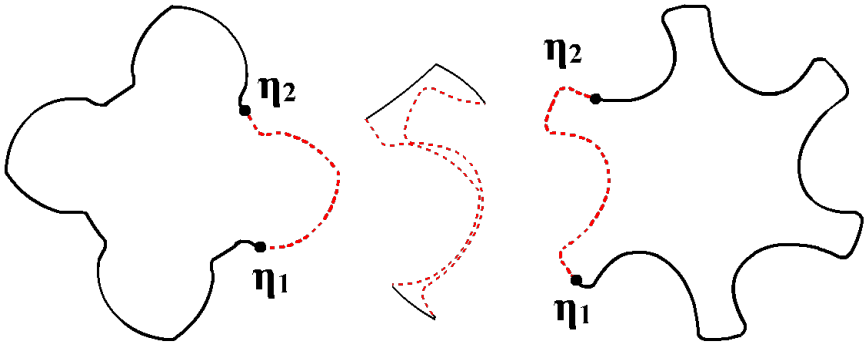


Figure 6.9: The exact rotor spline-fits are evaluated from η_1 to η_2 and the resulting point clouds are combined with the C-grid cuts

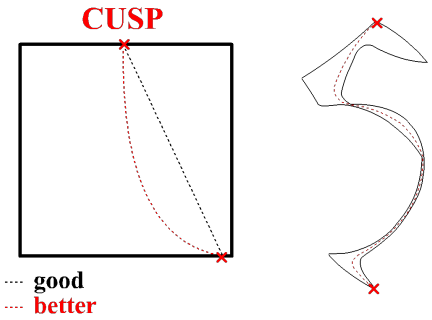


Figure 6.10: A splitting curve can be generated by traversing the parametric domain from one CUSP-point to the other. The properties of this splitting curve is tuned by changing the path taken. We choose the path such that the separator is split most-evenly at the narrow gaps.

tion function associated with the eastern boundary of the k-th slice. Upon completion of every L-th parameterization, we are in the possession of K/L singlepatch parameterizations for the separator as well as a reparameterization function η'_k, $k = 1, L, 2L, \ldots$ for every L-th slice. We build a global reparameterization function $\mathcal{N}'(\theta, \eta)$ by blending the available η'_k over the θ-interval (typically $[0, \pi/2]$). This ensures that the assignment of parametric values to the input point cloud is continuous in the rotational angle θ, which, in turn, results in mappings whose control points change as a smooth function of θ. The restriction of $\mathcal{N}'$ to $\theta = \theta_k$ then reparameterizes the input of the k-th slice.
We use the K/L available singlepatch parameterizations and interpolate them in θ. The inner control points of the resulting interpolation function are extracted and serve as an initial guess for the remaining slices, in a way similar to the rotor O-grids. This reduces the required amount of iterations, whereby the reduction depends on the density of the database.
Upon completion of all K-slices, we traverse $\hat{\Omega}$ from the preimage of one CUSP-point to the other, leaving splitting-curves in our wake (see Figure 6.10). Their properties are fully determined by the path taken. To maximize the quality of the parameterizations

in step (vi), we traverse $\hat{\Omega}$ such that the separator is split most-evenly on both sides of small gaps. Upon completion, we are in the possession of a splitting-curve for all discrete angles θ_k. As a last step, we utilize the resulting database to parameterize the separator with two patches, one on each side of the splitting curve. Hereby, we employ the same database-driven computational approach as for the singlepatch parameterized separator. To achieve element conformity between both sides of the separator and the C-grids, we employ the same knot vector(s) in the ξ-direction.

6.9. RESULTS

We have implemented the approach from Section 6.8 in the FEA Python-library *Nutils* [vZvZV+19]. The geometry has been parameterized for $K = 200$ discrete angles over the rotational interval $[0, \pi/2]$ (corresponding to a quarter rotation on the male rotor after which the initial position is again assumed). Figures 6.11 and 6.12 show the two rotor C-grids at $\theta = \theta_1$ and $\theta = \theta_{150}$, respectively. We used $L = 5$ to fill database with 40 sin-

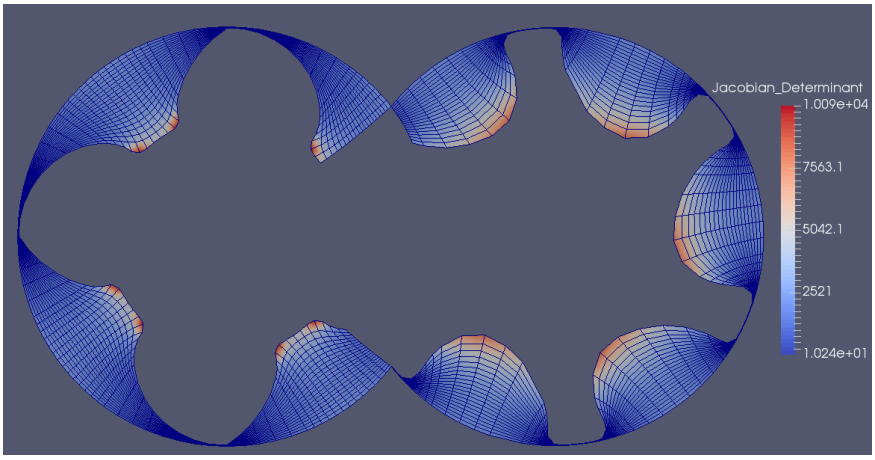


Figure 6.11: The two C-grids at $\theta = \theta_1$

glepatch parameterizations of the separator. The singlepatch parameterized separator along with the computed splitting-curve is plotted for $\theta = \theta_1$, $\theta = \theta_{75}$ and $\theta = \theta_{150}$ in Figure 6.13. Here, we do not plot the isolines for improved visibility. Figure 6.14 shows the final geometry parameterization at $\theta = \theta_{100}$ while Figure 6.16 (b) plots a zoom-in on the conforming separator showing the parametric properties by the splitting curve. Finally, Figure 6.15 shows the final geometry at $\theta = \theta_1$ and 6.16 (a) a zoom-in on the separator.
With $K = 200$, the computation of the rotor O-grids converges after 1 iteration as soon as enough slices for a 5-th order extrapolation are available. The 40 ($L = 5$) initial singlepatch parameterized separators typically converge within 3 iterations on the fine grid using the hierarchical approach from section 6.7. Upon completion, we use the partially-filled database to interpolate in θ. Taking the restriction to the current rotational angle as an initial guess, for the remaining slices, convergence is typically reached within $1-2$ iterations. The same level of efficiency is achieved for the two-patch parameterized sep-

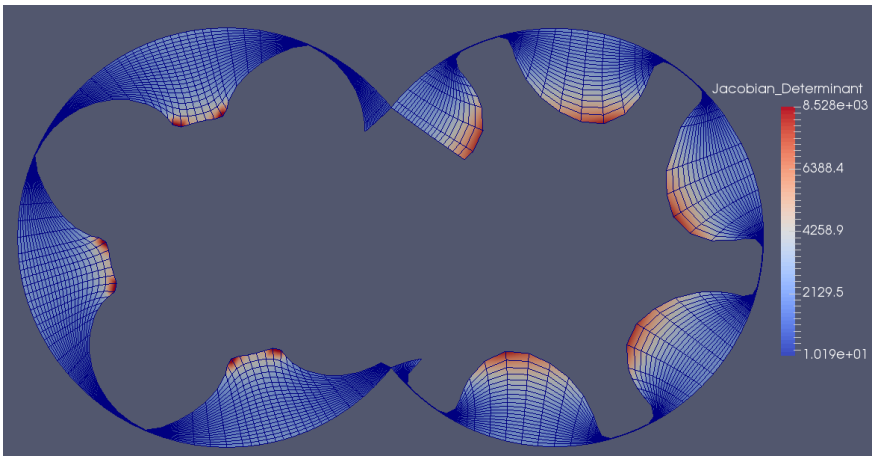


Figure 6.12: The two C-grids at $\theta = \theta_{150}$

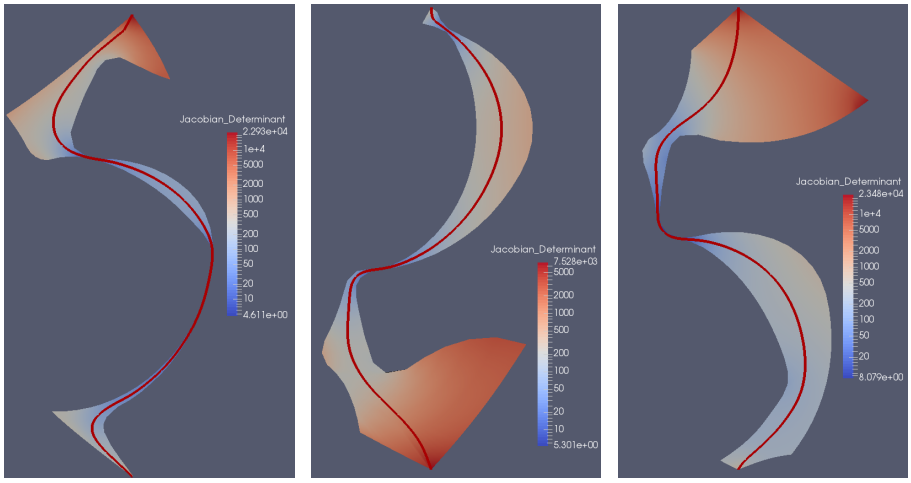
arator.

6.10. DISCUSSION

In Section 6.8, we presented a framework that employs the principles from Sections 6.4 to 6.7 for the fully automated parameterization of twin-screw machine geometries at every rotational angle θ_k, given no more than a description of the boundary contours. Here, a database-driven approach that performs inter- and extrapolation in θ reduces the required number iterations until convergence is achieved. The quality of the interpolation (and by that the expected number of required iterations) greatly depends on the ratio between K and L. In Section 6.9, we used $(K, L) = (200, 5)$. Here, we regard the choice $K = 200$ as realistic for an accurate numerical flow simulation. The interpolation-enabled reduction to $1-2$ iterations is a remarkable result since it reduces the computational costs to nearly the same level as algebraic parameterization techniques, while being more robust in practice.
The properties of the splitting curve, which are fully determined by the path taken in $\hat{\Omega}$, have a profound influence on the parametric properties of the two-patch parameterized separator, cf. Figure 6.16. In (a), we clearly see the steep inter-element angles at the splitting curve interface. In (b) this is less pronounced. Here, approximate halving of the separator leads to decent results. However, a different selection criterion may lead to more desirable outcomes and constitutes a topic for future research. Hereby, a good trade-off between the steepness of the inter-elements angles by both the splitting curve and C-grid interfaces may serve as a selection criterion.

6.10.1. SHAPE OPTIMIZATION

As stated in section 6.1, we are particularly interested in performing shape optimization on a variable pitch-function. Given a particular rotor profile input, the goal is to minimize some objective function over the three-tuple of shape parameters $\boldsymbol{\alpha}$ comprised of



(a) $\theta = \theta_1$ (b) $\theta = \theta_{75}$ (c) $\theta = \theta_{150}$

Figure 6.13: The singlepatch parameterized separator along with the generated splitting curve for various values of θ.

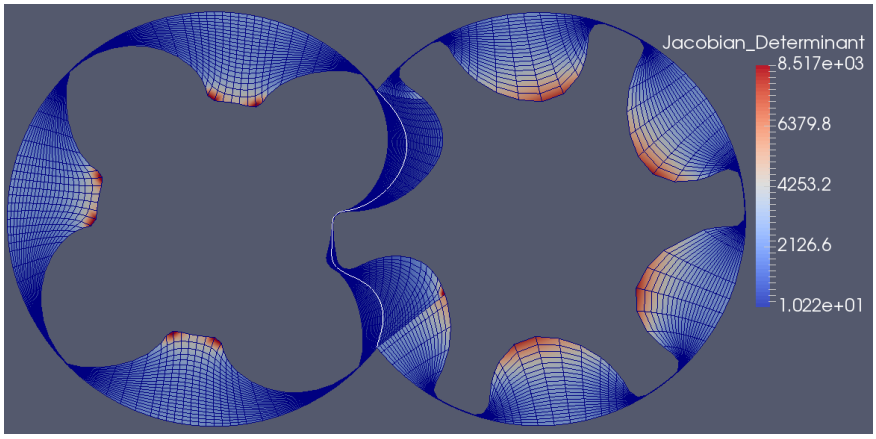


Figure 6.14: The two-patch parameterized separator along with the splitting curve at $\theta = \theta_{100}$

left- and right pitches $\dot{\theta}_{l,r}$ and the z-coordinate l_z at which the pitch changes. We base our approach on the observation that the planar slices at angle θ_k coincide with planar cross-sections of a volumetric parameterization in the z-direction.
The idea is to parameterize the geometry at a large number of discrete angles θ_k with angular increment $\Delta\theta$ and fill the database with a large number of planar parameterizations $\mathbf{x}_h^{\theta_k}$. A parameterization for a particular configuration of $\boldsymbol{\alpha} = \left(\dot{\theta}_l, \dot{\theta}_r, l_z\right)^T$ is accomplished by a proper stacking of the slices in the z-direction. Hereby, a (locally) stronger pitch requires a higher slice-density for an accurate description of the geometry, while

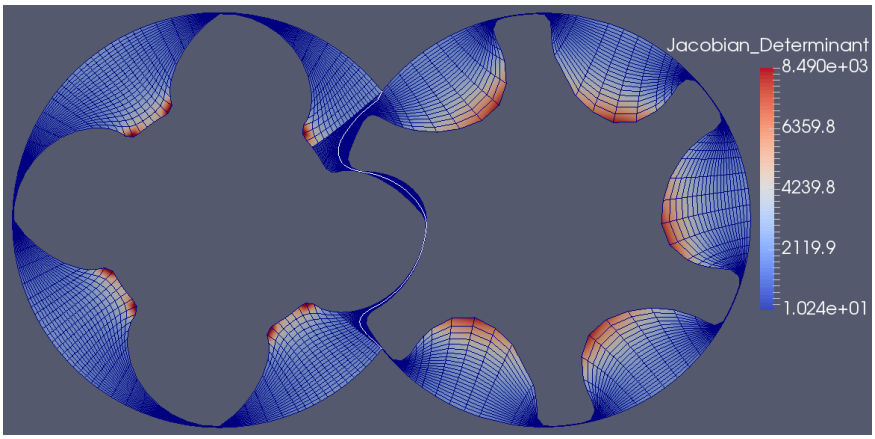
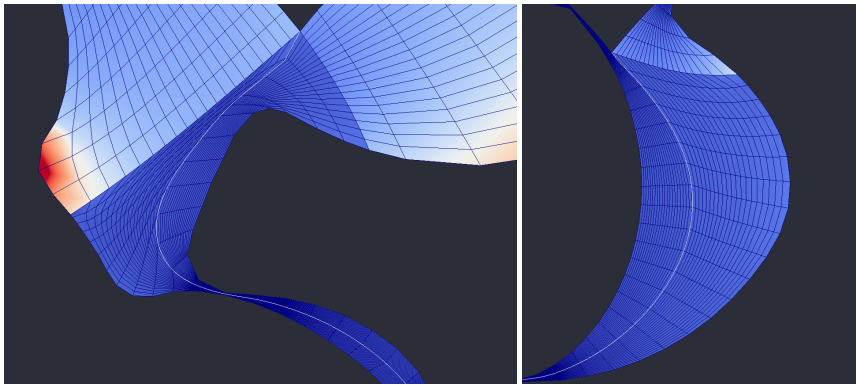


Figure 6.15: The two-patch parameterized separator along with the splitting curve at $\theta = \theta_1$



(a) $\theta = \theta_1$ (b) $\theta = \theta_{100}$

Figure 6.16: A zoom-in on the two-patch parameterized separator at $\theta = \theta_1$ (a) and $\theta = \theta_{100}$ (b).

a lower pitch allows for less slices. A database should therefore be generated with a slice-density that corresponds to the largest admissible pitch in the design space. Lower-pitched segments can be parameterized using a subset of the available slices $\mathbf{x}_h^\theta$. Since interpolation in the z-direction is a relatively cheap operation, a decent parameterization for a given $\boldsymbol{\alpha}$ comes at a relatively low cost.
Figure 6.17 shows a segment of a volumetric geometry with constant pitch, generated by the stacking of a large number of planar slices in the z-direction. Finally, figure 6.18 shows a segment of the separator with non-constant pitch along with a dotted red line to indicate the z-coordinate at which the pitch changes. This geometry has been constructed using the same planar slices as in Figure 6.17 but with a tighter stacking in the stronger-pitched region.

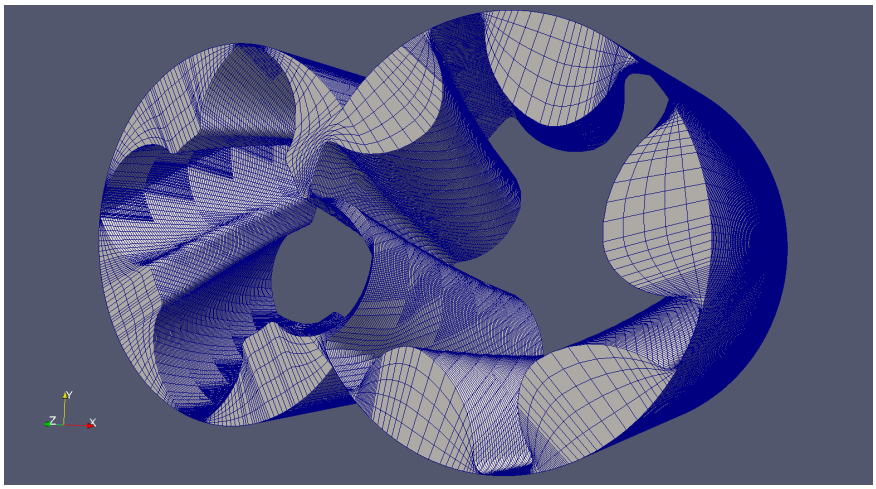


Figure 6.17: Part of a volumetric parameterization acquired by stacking a large number of planar slices.

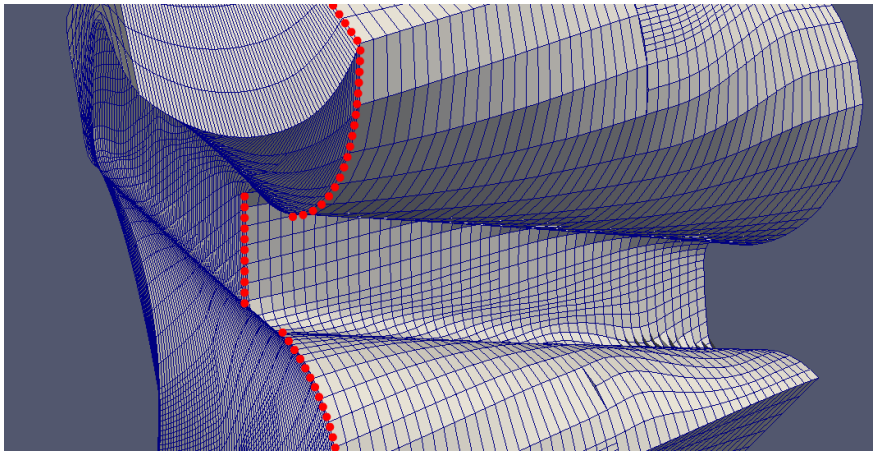


Figure 6.18: Part of a volumetric parameterization of the separator with non-constant pitch. The coordinate at which the pitch changes is indicated by the dotted red line.

6.11. CHAPTER CONCLUSIONS

In this chapter we presented a practical approach for the parameterization of twin-screw machine geometries with spline functions. For this, we adopted the principles of Elliptic Grid Generation and introduced a computational approach that is compatible with the principles of Isogeometric Analysis. We proposed automated boundary contour reparameterization techniques that further improve the quality of the resulting parameterization.

We have successfully applied the approach to a twin-screw machine geometry. We concluded that the parametric properties can be improved by optimizing the properties of

the splitting curve, a necessary ingredient for the parameterization of the separator. Finally, we have given a qualitative explanation of how the proposed techniques may be employed for performing database-driven shape optimization on a variable rotor pitch and presented an example of a volumetric parameterization resulting from stacking of a large number of planar slices in the z-direction.

REFERENCES

[ANS] Inc. ANSYS. Ansys cfx. https://www.ansys.com/products/fluids/ansys-cfx/. [Online; accessed 2018-03-24].

[Cas91] José E Castillo. *Mathematical aspects of numerical grid generation*. SIAM, 1991.

[Coo67] Steven A Coons. Surfaces for computer-aided design of space forms. Technical report, DTIC Document, 1967.

[Die95] Paul Dierckx. *Curve and surface fitting with splines*. Oxford University Press, 1995.

[FC80] Frederick N Fritsch and Ralph E Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, 17(2):238–246, 1980.

[GH73] William J Gordon and Charles A Hall. Construction of curvilinear coordinate systems and applications to mesh generation. *International Journal for Numerical Methods in Engineering*, 7(4):461–477, 1973.

[GT82] William J Gordon and Linda C Thiel. Transfinite mappings and their application to grid generation. *Applied Mathematics and Computation*, 10:171–233, 1982.

[HCB05] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *CMAME*, 194:4135–4195, 2005.

[HMV18a] J Hinz, M Möller, and C Vuik. Elliptic grid generation techniques in the framework of isogeometric analysis applications. *Computer Aided Geometric Design*, 2018.

[HMV18b] Jochen Hinz, Matthias Möller, and Cornelis Vuik. Spline-based parameterization techniques for twin-screw machine geometries. In *IOP Conference Series: Materials Science and Engineering*, volume 425, page 012030. IOP Publishing, 2018.

[SCO] SCORG. Screw compressor rotor grid generator. http://pdmanalysis.co.uk/scorg/. [Online; accessed 2018-03-24].

[TM] Twin-Mesh. Twin-mesh software. https://www.twinmesh.com/. [Online; accessed 2018-03-24].

[TSW98] Joe F Thompson, Bharat K Soni, and Nigel P Weatherill. *Handbook of grid generation*. CRC press, 1998.

[vZvZV+19] Gertjan van Zwieten, Joost van Zwieten, Clemens Verhoosel, Eivind Fonn, Timo van Opstal, and Wijnand Hoitinga. Nutils, June 2019.

7

Boundary-Conforming Finite Element Methods for Twin-Screw Extruders using Spline-Based Parameterization Techniques

This chapter is based on the publication from [HHME20]. It further develops the framework presented in Chapter 6 and applies it to co-rotating twin-screw machine extruders, a geometry of the same type as rotary-screw compressors. Basing the approach on the algorithm from Chapter 3 overcomes the need to find a *splitting-curve* which connects the two CUSP-points of the casing, cf. Chapter 6. Instead, this curve establishes itself as part of the solution to a PDE-problem, improving the efficiency and robustness of the framework, as well as aiding in parameterization quality.
By collocating the mapping operator, the spline-based framework forms the basis for generating classical meshes. They, in turn, are utilized for numerical simulation.
This work is the result of a collaboration with *J. Helmig and S. Elgeti* from RWTH Aachen University, Aachen, Germany and is divided into two parts. The first part develops the geometrical aspects, while the second part considers the simulation. Here, the simulation part (Section 7.3 onward) is the intellectual property of Helmig and Elgeti.

This chapter presents a novel spline-based meshing technique that allows for using boundary-conforming meshes in unsteady flow and temperature simulations of co-rotating twin-screw extruders. Spline-based parameterizations for a wide variety of extruder designs are generated using Elliptic Grid Generation. They are evaluated

in a number of discrete points to yield a coarse classical mesh. Here, the use of a special control mapping fine-tunes the properties of the coarse mesh, such as element orthogonality at the boundaries. The coarse mesh is used as a *scaffolding* to generate a boundary-conforming, finer mesh at run-time. This makes the method memory-efficient. Additionally, the adaptation at run-time is extremely cheap compared to computing the flow solution. Furthermore, the approach circumvents the need for expensive re-meshing and projecting solutions from grids at previous time-instants to the current, improving efficiency and accuracy. This methodology is incorporated into a space-time finite element framework. We present time-dependent test cases of non-Newtonian fluids for complex screw designs in two and three spatial dimensions. They demonstrate the potential of the method also for arbitrarily complex industrial applications.

7.1. INTRODUCTION

Co-rotating twin-screw extruders are widely-used devices in the plastic-producing industry. They are used to distribute and disperse polymers and additives since they provide short residence times and extensive mixing. Typical screw elements are conveying elements that are used to transport the plastic melt from the feed section towards the die, as well as kneading and mixing elements. The latter are specially-tailored to account for dispersive and distributive mixing.

Performing experiments to obtain detailed information about velocity, pressure, and temperature distribution is very complex, time-consuming, and expensive, if not impossible for twin-screw extruders. This makes Computational Fluid Dynamics (CFD) using finite element analysis an appealing tool for obtaining detailed information about the flow inside twin-screw extruders. However, numerical flow-analysis is not trivial due to constantly-moving domains and small gap sizes. These geometry-features make meshing extremely challenging. To tackle this problem, a wide range of approaches have been proposed.

One-dimensional models have been developed for predicting the pressure build-up [CW91] or fill length and specific mechanical energy [VVD98]. Full 3D results have been obtained using commercial software like POLYFLOW which is based on a mesh superposition method that avoids re-meshing [ZFCH09, ALA04]. A rather recent development are methods using smoothed particle hydrodynamics [RC18, WPE+18]. These methods are extremely useful when only partly filled extruders are considered.

Fictitious domain methods constitute another popular choice. Meshing is avoided since the actual geometry is embedded into a fixed background mesh. A classical example is the method presented in [VCDV09] that is used in [DDMN+14] to compare 3D results with 1D estimates. However, in [FHM+12] it has been shown that classic fictitious domain methods lack accuracy inside the small gap regions. This drawback can be mitigated by, for example, XFEM. Other examples of fictitious domain methods are the Body Conformal Enrichment method presented in [HI13] or a method that attempts to concentrate the background mesh at the screw interface using algebraic operations [MTH+14] . As a drawback, such methods require additional effort to describe the screw boundary exactly, as well as capturing the flow effects in the small gaps. Furthermore,

load balancing for highly parallel large-scale computations is extremely difficult.
A different approach are boundary-conforming methods. The mesh represents the geometry exactly in terms of the underlying finite element discretization. This allows for strongly imposing boundary conditions as well as constructing high-quality boundary-layer meshes. As a drawback, generating a time-dependent boundary-conforming mesh is challenging.
Thus, most simulations employ a *snapshot* technique. An individual mesh is generated for each screw orientation. This is a valid simplification since the flow inside the extruder can be considered to be quasi-steady. This approach has been applied in many publications, cf. [BHW00, MKGJ14]. Also, steady temperature results have been obtained in [IKF01, KM07]. However, these methods may be restrictive when simulating time-dependent quantities.
A continuous time-dependent mesh is needed. This requires efficient mesh-update techniques to avoid constant re-meshing and, in turn, expensive projections onto to the new mesh.
In [HBE18], we present the Snapping Reference Mesh Update Method (SRMUM), an efficient mesh update method for twin-screw extruders. It allows for computing time-dependent temperature results without the need for re-meshing. The method employs a structured background mesh that constantly adapts to the current screw configuration in a boundary-conforming manner. As a drawback, SRMUM is limited to convex screw geometries.
Instead, [HMV18a] generates analysis-suitable spline-based parameterizations. The resulting meshes are suitable for numerical analysis using Isogeometric Analysis (IGA) as introduced in [HCB05]. The approach is based on Elliptic Grid Generation, which can handle nonconvex screw designs. The PDE-based nature of the method makes it suitable for gradient-based shape optimization by simply adding the parameterization as a PDE-constraint to the shape optimization formulation, which can then be differentiated with respect to the design variables [HMV18b]. Furthermore, classical finite element meshes have been extracted from the spline parameterization and used to compute first flow results inside idealized twin-screw compressors [MH18].
In this work, we aim to adapt the general concept of spline-based meshing developed for twin-screw compressors to twin-screw extruders and combine it with certain aspects of SRMUM. This allows for generating flexible, high quality, time-dependent and FEM-suitable meshes.
The approach is based on taking cross sections through a volumetric extruder geometry leading to planar extruder contours at a large number of rotational angles θ_i (see Figure 7.1). A spline parameterization $\mathbf{x}_i$ is generated for the planar profiles at every discrete θ_i using Elliptic Grid Generation techniques.
The spline parameterizations are evaluated in a large number of points to generate a mesh *scaffolding* to adapt a SRMUM-like background mesh to the current screw shape via interpolation. A volumetric mesh is then built from the planar meshes by stacking them in the z-direction.
Once a spline parameterization has been generated, generating finer meshes for, e.g., refinement studies, is a cheap operation. Furthermore, function evaluations in the spline mappings $\mathbf{x}_i$ can be tuned using a *control mapping*, which accounts for special proper-

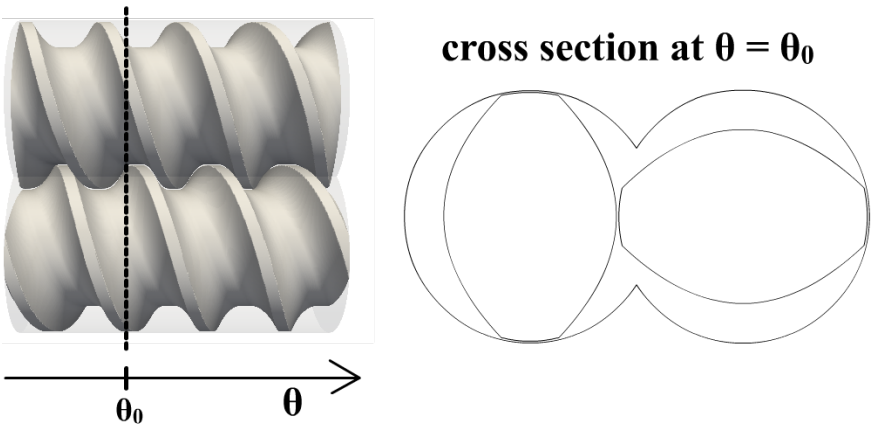


Figure 7.1: A volumetric mesh for an extruder geometry is built from a large number of planar cross sections parameterized in the plane using Elliptic Grid Generation with spline functions.

ties of the mesh, like orthogonality at the screw surface. Extracting only a mesh scaffolding prevents the need to save the full volumetric mesh for all possible time instances, which would be very time- and memory-consuming. Furthermore, the computational costs of updating the mesh at each time instant is negligible compared to computing the flow solution.

This chapter is structured as follows: Section 7.2 explains the meshing concept. This includes the generation of a spline parameterization up to the analysis-suitable mesh. The underlying physical equations that model the flow inside twin-screw extruders are discussed in Section 7.3, as well as the space-time finite element solution method. Section 7.4 contains two- and three-dimensional numerical examples that aim to validate as well as highlight the advantages of the presented method.

7.2. GRID GENERATION

Since the geometry pipeline provides no more than a description of the twin-screw profile (typically in the form of a point cloud), the first step towards numerical simulation is generating a planar computational mesh. For the mesh generation, we employ the following two-step approach: at every discrete angle θ_k, we first generate a spline-based geometry description which we then evaluate it in a large number of discrete points with known connectivity in order to yield a classical mesh. The advantages of a spline-based description is the possibility to fine-tune the properties of the mesh by appropriately choosing the evaluation abscissae, which follow from a precomputed control mapping. In the following, we give a brief introduction to spline surfaces after which we discuss the basics of the numerical algorithm employed in the meshing process and its applications to the target geometry.

7.2.1. B-SPLINES

As described in [PT12], B-splines are piecewise-polynomial functions that can be constructed so as to satisfy various continuity properties at the places where the polynomial segments connect. Their properties are determined by the entries of the so-called *knot vector*

$$\Xi = \{\xi_1, \xi_2, \ldots, \xi_{n+p+1}\}. \tag{7.1}$$

The knot vector is a nondecreasing sequence of parametric values $\xi_i \subset [0,1]$ that determine the boundaries of the segments on which the spline basis is polynomial. Selecting some polynomial order p, the p-th order spline-functions $N_{i,p}$ are constructed recursively, utilizing the relation (with $\frac{0}{0} \equiv 0$)

$$N_{i,s}(\xi) = \frac{\xi - \xi_i}{\xi_{i+1} - \xi_i} N_{i,s-1}(\xi) + \frac{\xi_{i+s+1} - \xi}{\xi_{i+s+1} - \xi_{i+1}} N_{i+1,s-1}(\xi), \tag{7.2}$$

starting from

$$N_{i,0} = \begin{cases} 1 & \text{if } \xi_i \le \xi < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases}, \tag{7.3}$$

and iterating until $s = p$. The support of basis function $N_{i,p}$ is given by the interval $\mathscr{I}_{i,p} = [\xi_i, \xi_{i+p+1}]$ and the amount of continuous derivatives across knot ξ_j is given by $p - m_j$, where m_j is the multiplicity of ξ_j in $\mathscr{I}_{i,p}$. In practice, $\xi_1 = 0$ is repeated $p+1$ times as well as ξ_{n+p+1} such that $\xi_1 = \ldots = \xi_{p+1} = 0$ and $\xi_{n+1} = \ldots = \xi_{n+p+1} = 1$. As a result, the corresponding basis $\sigma = \{N_{1,p}, \ldots, N_{n,p}\}$ forms a non-negative partition of unity on the entire parametric domain $[0,1]$, that is:

$$\sum_{i=1}^{n} N_{i,p}(\xi) = 1, \quad \text{with} \quad N_{i,p}(\xi) \ge 0, \tag{7.4}$$

for all spline functions $N_{i,p}$ [HCB05]. Figure 7.2 shows the $p = 3$ B-spline basis resulting from the knot vector

$$\Xi = \left\{0,0,0,0, \tfrac{1}{7}, \tfrac{2}{7}, \tfrac{3}{7}, \tfrac{4}{7}, \tfrac{5}{7}, \tfrac{6}{7}, 1,1,1,1\right\}. \tag{7.5}$$

The extension to bivariate spline bases is now straightforward: given two univariate bases $\sigma_\Xi = \{N_{1,p}, \ldots, N_{n,p}\}$ and $\sigma_\mathscr{H} = \{M_{1,q}, \ldots M_{m,q}\}$, we build a bivariate spline space $\mathscr{V}_h$ with basis $[\mathscr{V}_h] = \{w_{i,j} \mid (i,j) \in \{1,\ldots,n\} \times \{1,\ldots,m\}\}$, by means of a tensor-product, where

$$w_{i,j}(\xi, \eta) = N_{i,p}(\xi) M_{j,q}(\eta). \tag{7.6}$$

The values contained in Ξ and $\mathscr{H}$ (without knot repetitions) hereby become the boundaries of the polynomial segments, which can be regarded as the counterparts of classical elements.

We construct the mapping of a B-spline surface as follows:

$$\mathbf{x} = \sum_i \sum_j \mathbf{c}_{i,j} w_{i,j}, \tag{7.7}$$

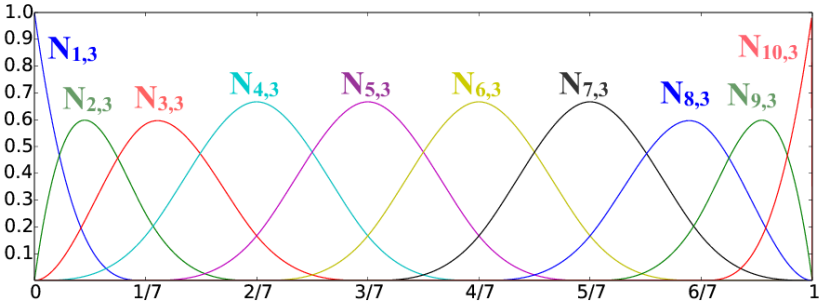


Figure 7.2: The univariate B-spline basis corresponding to the knot vector Ξ from (7.5).

where the $\mathbf{c}_{i,j} \in \mathbb{R}^2$ are referred to as the *control points*. We refer to the $\mathbf{c}_{i,j}$ with $i \in \{1, n\}$ or $j \in \{1, m\}$ as the *boundary control points* while the remaining $\mathbf{c}_{i,j}$ are called *inner control points*. Bluntly put, the local density of spline basis functions is determined by the local number of knots. As such, a cleverly chosen knot vector can be utilized to properly resolve important features of the geometry. Furthermore, as $\mathbf{x} \in \mathcal{V}^2$ is a linear combination of the $w_{i,j} \in \mathcal{V}_h$, it will inherit the local continuity properties of the basis. Therefore, many geometrical features can be better captured by a clever choice of the knot multiplicities in Ξ and $\mathscr{H}$.

7.2.2. SPLINE-BASED MESHING TECHNIQUES

In the following, we shall drop the tensor-product indexing used in (7.7) and replace it by a single global index. As such, the mapping operator will be of the form

$$\mathbf{x} = \sum_{i \in \mathscr{I}_I} \mathbf{c}_i w_i + \sum_{j \in \mathscr{I}_B} \mathbf{c}_j w_j, \tag{7.8}$$

where $\mathscr{I}_I$ and $\mathscr{I}_B = \{1, \dots, N\} \setminus \mathscr{I}_I$ refer to the index-sets of inner and boundary basis functions, respectively. The mapping $\mathbf{x}$ is a function from the parameter domain $\hat{\Omega} = [0,1]^2$ to the physical geometry Ω, whose boundaries $\partial\Omega$ follow from a regression of the employed spline space with basis $[\mathcal{V}_h] = \{w_1, \dots, w_N\}$, resulting from the tensor-product knot vector $\boldsymbol{\Xi} = \Xi \times \mathscr{H}$, to the input point cloud.

For the proper selection of $\boldsymbol{\Xi}$ and the subsequent regression, we utilize the approach illustrated in Section 6.5. Our methodology consists of a stabilized least-squares fit of the points against the basis and an adaptive re-selection of $\boldsymbol{\Xi}$ based on the local magnitude of the projection residual. The process is repeated until the residual is deemed sufficiently small. Upon completion, the restriction of $\mathbf{x}$ to $\partial\hat{\Omega}$ parameterizes $\partial\Omega$. Note that the above procedure essentially selects the boundary control points in (7.8) such that $\mathbf{x}|_{\partial\hat{\Omega}} = \partial\Omega$ is a regression of the input point cloud (hence the word *boundary control points*). However, the inner control points are unknown at this stage. As such, the task of any parameterization algorithm is to properly select the $\mathbf{c}_i$ in (7.8), such that the resulting parameterization is folding-free.

A computationally inexpensive, yet often sufficiently powerful method is transfinite interpolation [GH73]. The inner control points follow from an interpolation of the south-

ern and northern boundaries in combination with an interpolation from east to west. Let $\boldsymbol{\gamma}_e, \boldsymbol{\gamma}_w, \boldsymbol{\gamma}_s$ and $\boldsymbol{\gamma}_n$ parameterize the four segments of $\partial\Omega$. In a spline-based setting, the mapping operator is constructed as follows:

$$\begin{aligned}\mathbf{x}(\xi,\eta) = (1-\xi)\boldsymbol{\gamma}_w(\eta) + \xi\boldsymbol{\gamma}_e(\eta) + (1-\eta)\boldsymbol{\gamma}_s(\xi) + \eta\boldsymbol{\gamma}_n(\xi) \\ - (1-\xi)(1-\eta)\mathbf{p}_{0,0} - \xi\eta\mathbf{p}_{1,1} - \xi(1-\eta)\mathbf{p}_{1,0} - (1-\xi)\eta\mathbf{p}_{0,1},\end{aligned} \tag{7.9}$$

where the $\mathbf{p}_{i,j}$ denote the corners at $\mathbf{x}(i,j)$. The symbolic parameterization from (7.9) constitutes a recipe for determining the $\mathbf{c}_i$ in (7.8). They can be computationally inexpensively determined by a L_2-projection or a regression over the *Greville-abscissae* [Joh05] corresponding to $\boldsymbol{\Xi}$. Whenever $\mathbf{x}$ is an O-type parameterization, we simply remove the nonexistent pair of boundary contours ($\boldsymbol{\gamma}_n, \boldsymbol{\gamma}_s$ or $\boldsymbol{\gamma}_w, \boldsymbol{\gamma}_e$) from (7.9), as well as the $\mathbf{p}_{i,j}$ and perform unidirectional interpolation.
Transfinite interpolation is an $\mathcal{O}(N)$ operation but does not guarantee a folding-free mapping. In order to handle the complex characteristics of twin-screw extruders, it is desirable to have a more powerful parameterization technique in our arsenal.
A second class of parameterization methods are PDE-based, notably approaches based on the principles of *Elliptic Grid Generation* (EGG). The main purpose of EGG is to compute a mapping $\mathbf{x}: \hat{\Omega} \to \Omega$ such that the components of $\mathbf{x}^{-1}: \Omega \to \hat{\Omega}$ are harmonic in Ω. This is accomplished by imposing the following system of PDEs on $\mathbf{x}$:

$$g_{22}\mathbf{x}_{\xi\xi} - 2g_{12}\mathbf{x}_{\xi\eta} + g_{11}\mathbf{x}_{\eta\eta} = 0, \quad \text{s.t. } \mathbf{x}|_{\partial\hat{\Omega}} = \partial\Omega, \tag{7.10}$$

where the g_{ij} denote the entries of the metric tensor corresponding to $\mathbf{x}$. We have

$$\begin{pmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_\xi \cdot \mathbf{x}_\xi & \mathbf{x}_\xi \cdot \mathbf{x}_\eta \\ \mathbf{x}_\eta \cdot \mathbf{x}_\xi & \mathbf{x}_\eta \cdot \mathbf{x}_\eta \end{pmatrix}. \tag{7.11}$$

A justification of this approach is based on the observation that if $\partial\Omega$ satisfies certain regularity conditions, $\mathbf{x}$ will be bijective (and hence, folding-free) [Aza09]. As such, for analysis-suitability, we compute a sufficiently accurate approximation $\mathbf{x}_h$ of $\mathbf{x}$.

7.2.3. NUMERICAL IMPLEMENTATION

A basic numerical algorithm to approximately solve (7.10) with globally $C^{\geq 1}$-continuous spline bases is discussed in Chapter 2. Here, we follow a similar approach with auxiliary variables, as introduced in Chapter 3, whose purpose shall become apparent in Section 7.2.4. We approximately solve (7.10) with a mixed-FEM approach, introducing the auxiliary variable $\mathbf{x}_\xi \to \mathbf{u}$. This reduces the continuity requirements of the basis from globally $C^{\geq 1}$ to $C^{\geq 0}$ in ξ-direction.
Let

$$\mathbf{u}_h(\xi,\eta) = \sum_{i=1}^{\bar{N}} \mathbf{d}_i \bar{w}_i(\xi,\eta). \tag{7.12}$$

where the $\bar{w}_i$ are the canonical basis functions of the auxiliary spline space $\bar{\mathcal{V}}_h$.
In order to discretize (7.10), we express the ξ-derivatives in terms of $\mathbf{u}_h$ and introduce a

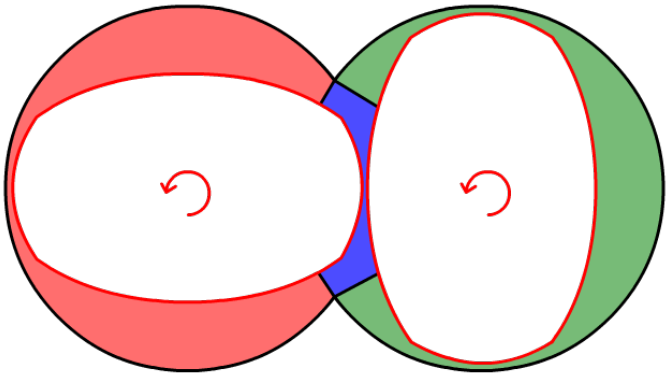


Figure 7.3: The topology we use for the parameterization of the geometry. Here, black boundaries are held fixed while red boundaries slide along the grid. The two C-grids (red, green) are parameterized (if possible) with transfinite interpolation while we use EGG (see (7.14)) for the separator (blue).

scaling. Consider

$$\mathscr{U}(\mathbf{u},\mathbf{x}) = \frac{g_{22}\mathbf{u}_\xi - g_{12}\mathbf{u}_\eta - g_{12}\mathbf{x}_{\xi\eta} + g_{11}\mathbf{x}_{\eta\eta}}{\underbrace{g_{11}+g_{22}}_{\geq 0} + \epsilon}, \tag{7.13}$$

with $\epsilon = 10^{-4}$ a small term that serves numerical stability. To compute an approximation $\mathbf{x}_h$ of $\mathbf{x}$, we solve the system

$$\begin{aligned} &\text{find } (\mathbf{u}_h,\mathbf{x}_h) \in \bar{\mathcal{V}}_h^2 \times \mathcal{V}_h^2 \quad \text{s.t.} \\ &\int_{\hat{\Omega}} \boldsymbol{\phi}_h \cdot \left(\mathbf{u} - \partial_\xi \mathbf{x}_h\right) \mathrm{d}\boldsymbol{\xi} + \int_{\hat{\Omega}} \boldsymbol{\sigma}_h \cdot U(\mathbf{u}_h,\mathbf{x}_h)\mathrm{d}\boldsymbol{\xi} = 0, \quad \forall (\boldsymbol{\phi}_h,\boldsymbol{\sigma}_h) \in \bar{\mathcal{V}}_h^2 \times (\mathcal{V}_h^\circ)^2 \\ &\text{and} \quad \mathbf{x}_h|_{\partial\hat{\Omega}} = \partial\Omega, \end{aligned} \tag{7.14}$$

where $\mathcal{V}_h^\circ \subset \mathcal{V}_h$ is the subset of vanishing functions on $\partial\hat{\Omega}$, spanned by the w_j in (7.8). This yields the control points $\mathbf{c}_i$ in (7.8) and $\mathbf{d}_i$ in (7.12). For a memory-efficient algorithm to tackle the root-finding problem (7.14), we refer to Chapter 3.
Upon solving (7.14), we are in the possession of $\mathbf{x}_h$ and $\mathbf{u}_h$. Here, $\mathbf{x}_h$ parameterizes Ω and $\mathbf{u}_h$ serves no further purpose and can be discarded. Unlike the exact solution of (7.10), its numerical approximation may fold due to numerical inaccuracies. Since a defect is a direct cause of insufficient numerical accuracy, it can be repaired by increasing the local density of spline functions wherever the defect is located and recomputing the mapping operator from the enriched spline space. In practice, we have observed folding to be a rare occurrence.

7.2.4. APPLICATIONS TO THE GEOMETRY

For the parameterization of the main geometry, we choose the topology illustrated in Figure 7.3. The topology consists of two C-type parameterizations connected by a third

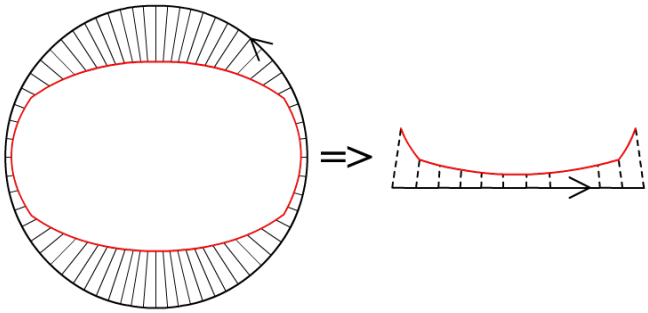


Figure 7.4: The first step towards a three-patch parameterization is finding a valid parameterization of a single rotor-casing O-grid. A valid parameterization is one in which vertical isolines do not cross upon a pullback of the circular casing onto a reference line segment.

parameterization (blue), henceforth referred to as the *separator*. The topology is designed such that upon rotation, the rotor lobes slide along the grid in the parametric domain while the grid is held fixed at the casings. Furthermore, the casings are held arc-length parameterized while the parametric properties of the rotor lobes depend on the rotation angle.

As such, the first step in building a three-patch parameterization is finding a valid O-grid parameterization of a single rotor plus casing. We take the circular casing and map it onto a straight line segment. A valid parameterization is one in which the vertical isolines do not intersect under the mapping (see Figure 7.4). To find a valid lobe parameterization, we employ the matching algorithm from Section 2.7 to the rotor and casing point clouds. The points are matched based on their Euclidean distance in a hierarchical fashion and matched points receive the same parametric value. Upon completion, we have a monotone reparameterization function $\xi' : [0,1] \to [0,1]$ and we assign the parametric value $\xi_i = \xi'(\hat{\xi}_i)$ to the i-th point in the rotor point cloud. Here, the $\hat{\xi}_i$ result from a chord length parameterization of the point cloud. The steps described above are carried out for $\theta = 0$ only. The reparameterization function can be reused for different rotational angles by performing an appropriate periodic shift that is based on the value of θ.

The coordinates of the points along with their parametric values are utilized for a regression of the left and right rotor O-grids to a suitable spline basis. Here, the knot vector is chosen such that the regression residual is below a user-defined threshold (see Section 6.5). Upon completion, the boundary curves are known and we utilize interpolation (see (7.9)) to form folding-free O-grid parameterizations for both rotor casing pairs.

Remark. In case finding a valid O-grid parameterization through interpolation fails, we can fall back on EGG, as in Chapter 6.

Next, we cut the left and right rotor O-grids at the parametric values that correspond to the casing CUSP points to form the left and right rotor C-grids.

We repeat above steps for a large number of discrete angles $\Theta = \{\theta_1, \ldots, \theta_K\}$ taken from the interval that corresponds to a rotation after which the initial position is again ob-

tained (typically the interval $[0,\pi]$). Upon completion, our database is filled with C-grid parameterizations for all discrete angles Θ. We combine the C-grid boundaries along with the rotor lobe parts that were cut from the rotor-casing O-grids in order to generate one boundary description of the separator, as shown in Figure 7.3. Clearly, a spline regression of the northern and southern boundaries has to be performed with a locally C^0-continuous spline basis, due to the spiked nature of the input point clouds. Therefore, we assign the parametric value $\xi = 0.5$ to the CUSP-points and utilize a knot vector with p-fold internal knot repetition at $\xi = 0.5$ for the ξ-direction.
The next step is assigning suitable parametric values to the points contained in the eastern and western input point clouds. Unfortunately, we have observed that chord length parameterization leads to unsatisfactory results. In order to ensure that similar parametric values are assumed on either sides of small gaps, we again apply the matching algorithm from Section 2.7. In contrast to the C-grid case, we do not keep one side chord length parameterized but let both sides float. If two points p_i^l and p_j^r have been matched, we assign the parametric value

$$\eta = \tfrac{1}{2}(\hat{\eta}_i^l + \hat{\eta}_j^r)$$

to both of them. Here, the values of $\hat{\eta}_i^l$ and $\hat{\eta}_j^r$ correspond to chord length parameterizations. Upon completion, we have two reparameterization functions that we utilize to assign parametric values to the input point clouds, similar to the C-grid case.
We perform the above steps for each L-th discrete angle $\theta_k \in \Theta$. Typically we take $L = 5$. A global reparameterization function is constructed by blending each L-th reparameterization function over the θ-interval. Blending enables achieving smoothness in the parametric properties of the separator contours as a function of θ, which would be lost if we reparameterized at every θ_k, due to the discrete nature of the reparameterization algorithm.
Upon completion, we start filling our database with separator parameterizations utilizing the EGG algorithm from (7.14). The auxiliary spline space $\bar{\mathcal{V}}_h$ is of Raviart-Thomas type [BdFS] on each of the macro elements [BS13] $\boldsymbol{\xi} \in [0,0.5]\times[0,1]$ and $\boldsymbol{\xi} \in [0.5,1]\times[0,1]$, with C^0 interface coupling. To be precise, let

$$[\mathcal{V}_h] = \sigma_{\Xi}^{p_1} \times \sigma_{\mathscr{H}}^{p_2} \quad \text{and} \quad [\bar{\mathcal{V}}_h] = \sigma_{\bar{\Xi}}^{q_1} \times \sigma_{\bar{\mathscr{H}}}^{q_2} \tag{7.15}$$

be built from the knot vectors

$$\Xi^{p_1,p_2} = \Xi^{p_1} \times \mathscr{H}^{p_2} \quad \text{and} \quad \bar{\Xi}^{q_1,q_2} = \bar{\Xi}^{q_1} \times \bar{\mathscr{H}}^{q_2}, \tag{7.16}$$

respectively. Here, the p_i and q_i denote the polynomial orders used in each direction. If

$$\Xi^{p_1} = \{ \underbrace{0,\dots,0}_{p_1+1 \text{ times}}, \xi_1,\dots,\xi_q, \underbrace{0.5,\dots,0.5}_{p_1 \text{ times}}, \xi_r,\dots,\xi_s, \underbrace{1,\dots,1}_{p_1+1 \text{ times}} \}, \tag{7.17}$$

we take $\bar{\Xi}^{q_1,q_2} = \bar{\Xi}^{p_1+1} \times \mathscr{H}^{p_2}$, with

$$\bar{\Xi}^{p_1+1} = \{ \underbrace{0,\dots,0}_{p_1+2 \text{ times}}, \xi_1,\dots,\xi_q, \underbrace{0.5,\dots,0.5}_{p_1+1 \text{ times}}, \xi_r,\dots,\xi_s, \underbrace{1,\dots,1}_{p_1+2 \text{ times}} \}. \tag{7.18}$$

In our examples, $[\mathcal{V}_h]$ is bicubic, i.e., $p_1 = p_2 = 3$.
The root-finding problem is initialized with a transfinite mapping (see (7.9)). Convergence can be further accelerated using multigrid techniques (see Chapter 6). We fill our database in a hierarchical fashion. As soon as the database contains separator parameterizations for a sufficiently large subset of angles from Θ, we perform database interpolation in order to generate initial guesses for the remaining angles to further speed up the process.
After the algorithm has finished, we have a three-patch parameterization for the geometry at every $\theta_k \in \Theta$. A classical mesh can be generated by evaluating the spline mapping in a large number of uniformly-spaced points. The properties of the mesh can be further tuned by using a control mapping. This particularly benefits the mesh quality in the separator. By the tuple $(\mathbf{x}_k, \mathbf{s}_k)$, we denote the mapping operator and the corresponding control mapping for the separator at $\theta = \theta_k$. The control mapping $\mathbf{s}_k(\mu,\nu) \in \tilde{\mathcal{V}}_h^2$ is a function from the unit square onto itself. Typically, we take $\left[\tilde{\mathcal{V}}_h\right]$ coarser than $[\mathcal{V}_h]$. The control points with respect to $\left[\tilde{\mathcal{V}}_h\right]$ follow from solving the following minimization problem:

$$\frac{1}{2}\int_{[0,1]^2}\left(\frac{\partial(\mathbf{x}_k\circ\mathbf{s}_k)}{\partial\mu}\cdot\frac{\partial(\mathbf{x}_k\circ\mathbf{s}_k)}{\partial\nu}\right)^2 \mathrm{d}\mu\mathrm{d}\nu \to \min_{\mathbf{s}_k\in\tilde{\mathcal{V}}_h^2}, \tag{7.19}$$

subject to boundary conditions. The cost function from (7.19) maximizes orthogonality in the composite mapping $\mathbf{x}_k \circ \mathbf{s}_k$. The integral is approximated with a collocation technique. The boundary conditions follow from the requirement that $\mathbf{s}_k$ be a mapping from the unit square onto itself. Here, we only let $\mathbf{s}_k$ slide in η-direction (see Figure 7.5) as this leads to better results for tube-like shaped geometries such as the separator. A sufficient linear condition for the aforementioned requirement is easily formulated and imposed as a constraint on (7.19). The minimization problem is tackled with an SLSQP [Kra88] algorithm while the gradient of the cost function is approximated with a colored finite difference approach. To speed up convergence, control mappings of previous slices can be utilized to initialize the next control mapping. We compute a control mapping for the separator at all discrete angles $\theta_k \in \Theta$. We do not compute control mappings for the C-grids (i.e., the control mapping is the identity). Upon completion, we build a mesh *scaffolding* for the separator from the composite mapping $\mathbf{x}_k \circ \mathbf{s}_k$ by evaluating it in a large number of uniformly-spaced points. By $n_{\mu,\nu}$, we denote the number of elements in μ and ν direction, respectively. An adaptive background mesh is generated by adding additional vertices via linear interpolation within the mesh scaffolding point cloud. The vertex positions are computed on the fly and do therefore not have to be stored explicitly (see Figure 7.6). For the background mesh, we denote the number of elements in radial and screw direction by n_r and n_s with $n_r \geq n_\mu$ and $n_s \geq n_\nu$, respectively. The ratios $n_r/n_\mu \geq 1$ and $n_s/n_\nu \geq 1$ are hence based on a trade off between memory requirements and grid quality, where ratios closer to 1 typically lead to better meshes (for a given number of elements) while increasing the memory requirements. The converse holds for larger ratios, whereby it is important to take $n_{\mu,\nu}$ sufficiently large so that the resulting mesh scaffolding collocates the folding-free $\mathbf{x}_k \circ \mathbf{s}_k$ sufficiently accurately and therefore does not fold itself. For the C-grids, the mesh scaffolding cannot fold due to an insufficient number of elements in μ-direction, thanks to the precomputed repa-

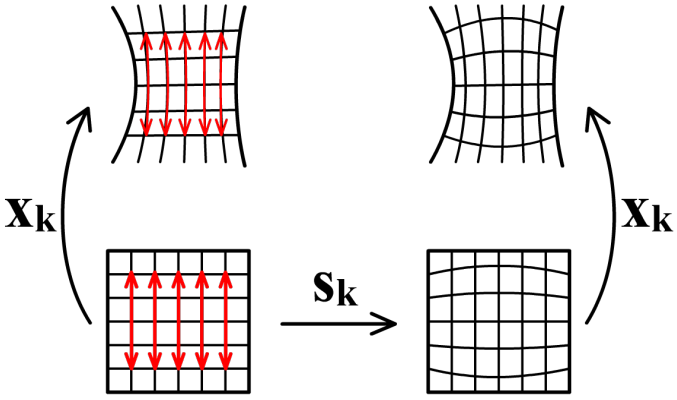


Figure 7.5: The control mapping serves the purpose of altering the parametric properties of the composite mapping. Here, we have chosen to orthogonalize the grid lines.

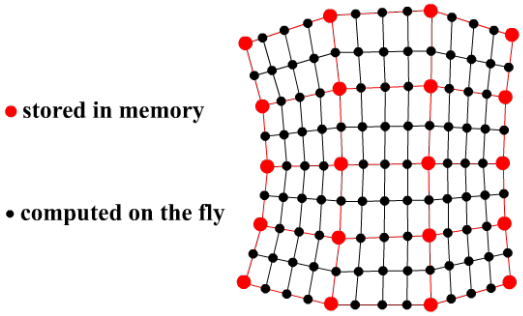


Figure 7.6: The composite mapping $\mathbf{x}_k \circ \mathbf{s}_k$ is evaluated in a number of uniformly-spaced abscissae and the resulting points (red) serve as a scaffolding for the remaining mesh vertices. Hereby it is important to make a good trade off between memory requirements and grid quality (more evaluation points tend to increase grid quality).

rameterization function (see Figure 7.4). Hence, we take $n_\mu = 1$ and add vertices until $n_r(C) = n_r(\text{separator})/2$ to yield a boundary-conforming mesh on both sides of the CUSP points. The storage requirements for the C-grids are thus fully determined by n_v, whose value is chosen to properly resolve the rotor boundaries. Typically, for the C-grids, we simply take $n_s = n_v$.

In order to construct a 3D mesh, we linearly connect the 2D meshes, where n_a determines the number of elements in z-direction. Every slice in z-direction knows its initial angle, which is used to interpolate between the stacked planar slices of the scaffolding.

7.3. GOVERNING EQUATIONS AND SOLUTION METHOD

7.3.1. GOVERNING EQUATIONS FOR FLOW AND TEMPERATURE OF PLASTIC MELT

We choose to represent the molten polymer in the extruder as viscous, incompressible and temperature-dependent fluid. A time-dependent computational domain, denoted by $\Omega_t \subset \mathbb{R}^{n_{sd}}$, is considered. It is enclosed by its boundary Γ_t, where $t \in (0,T)$ is an instance of time and n_{sd} the number of space dimensions. The velocity $\mathbf{u}(\mathbf{x},t)$ and pressure $p(\mathbf{x},t)$ are governed by the incompressible Navier-Stokes equations:

$$\nabla\cdot\mathbf{u} = 0 \quad \text{on } \Omega_t, \quad \forall\, t \in (0,T), \tag{7.20}$$

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) - \nabla\cdot\boldsymbol{\sigma} = \mathbf{0} \quad \text{on } \Omega_t, \quad \forall\, t \in (0,T), \tag{7.21}$$

where ρ is the fluid density. The stress tensor $\boldsymbol{\sigma}$ is used to close the set of equations:

$$\boldsymbol{\sigma}(\boldsymbol{u},p) = -p\boldsymbol{I} + 2\eta\left(\dot{\gamma},T\right)\boldsymbol{\varepsilon}(\boldsymbol{u}), \tag{7.22}$$

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}\left(\nabla\boldsymbol{u} + \left(\nabla\boldsymbol{u}^T\right)\right), \tag{7.23}$$

with η being the dynamic viscosity. It is constant for a Newtonian fluid and varies for Generalized Newtonian models with respect to temperature T and shear rate $\dot{\gamma}$.
Using single-phase Navier-Stokes equations implies the assumption that the extruder is fully filled. However, in practice it might occur that the flow channels inside the twin-screw extruder are only partially filled. In order to account for this, one would have to extend the model to also include the air phase, which results in solving a multi-phase flow problem. Popular numerical methods for multi-phase flow are the levelset [OS88] or the volume of fluid method [HN81]. For simplicity, only single-phase flow will be considered within this work.
The temperature $T(\mathbf{x},t)$ inside the extruder is governed by the heat equation:

$$\rho c_p\left(\frac{\partial T}{\partial t} + \boldsymbol{u}\cdot\nabla T\right) - \kappa\boldsymbol{\Delta}T - 2\eta\nabla\boldsymbol{u}\colon\boldsymbol{\varepsilon}(\boldsymbol{u}) = 0 \quad \text{on } \Omega_t, \quad \forall\, t \in (0,T). \tag{7.24}$$

The Dirichlet and Neumann boundary conditions for temperature and flow are defined as:

$$\mathbf{u} = \mathbf{g}^f \text{ on } (\Gamma_t)^f_g, \tag{7.25}$$

$$\mathbf{n}\cdot\boldsymbol{\sigma} = \mathbf{h}^f \text{ on } (\Gamma_t)^f_h, \tag{7.26}$$

$$T = g^T \text{ on } (\Gamma_t)^T_g, \tag{7.27}$$

$$\mathbf{n}\cdot\kappa\nabla T = h^T \text{ on } (\Gamma_t)^T_h. \tag{7.28}$$

$(\Gamma_t)^i_g$ and $(\Gamma_t)^i_h$ are complementary portions of Γ^i_t, with $i = f$ (Fluid), T (Temperature).
It is of course true that plastic melts exhibit viscoelastic behavior. We assume, however, that this is negligible in this context, since the residence time in the twin-screw

extruder is small. We use Generalized Newtonian models to account for shear-thinning and shear-thickening behavior of the polymer melt. Using Generalized Newtonian models implies that the viscosity depends on the invariants of the rate of strain tensor $\boldsymbol{\varepsilon}$, such as the shear rate $\dot{\gamma}$:

$$\dot{\gamma} = \sqrt{2\boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{u})}. \tag{7.29}$$

Within this work we use two different models, namely the Carreau and the Cross-WLF model.
The Carreau model is a very popular shear-thinning model in the plastic community [CDK79]. It is defined as:

$$\eta(\dot{\gamma}) = \eta_\infty + (\eta_0 - \eta_\infty)\left(1 + (\lambda\dot{\gamma})^2\right)^{\frac{n-1}{2}}, \tag{7.30}$$

where λ is the relaxation time, n is the power index, η_0 is the viscosity at zero shear rate and η_∞ is the viscosity at infinite shear rate.
The Cross-WLF model considers along with the effects of the shear rate also the influence of temperature on the viscosity [RO14]. We neglect the infinite viscosity such that the Cross model is defined as:

$$\eta(\dot{\gamma}, T) = \frac{\eta_0(T)}{1 + \left(\frac{\eta_0(T)\dot{\gamma}}{\tau^*}\right)^{(1-n)}}, \tag{7.31}$$

where τ^* is the critical shear stress at the transition from the Newtonian plateau. $\eta(\dot{\gamma}, T)$ depends on the temperature now. This relation is modeled via the WLF equation:

$$\eta_0(T) = D_1\, exp\left(-\frac{A_1(T - T_{ref})}{A_2 + (T - T_{ref})}\right), \tag{7.32}$$

with D_1 being the viscosity at a reference temperature T_{ref} and A_1 and A_2 are parameters that describe the temperature dependency.

Remark. In most of the available literature, Stokes equations are used to model the fluid inside a twin-screw extruder. However, Reynolds numbers around 0.1 occur in our applications s.t. the Stokes equations are not necessarily a valid assumption any more. Thus, we model the fluid using the Navier-Stokes equations which also include the advective term. It is noteworthy that the nonlinearity introduced due to the advective term is small compared to the one introduced by using Generalized Newtonian models.

7.3.2. SPACE-TIME FINITE ELEMENT DISCRETIZATION

We need to discretize the equations in space and time in order to capture the transient fluid flow. Additionally, the rotating screws introduce a constantly moving and deforming domain. A natural approach accounting for all of these requirements is the DSD/SST (Deforming Spatial Domain / Stabilized Space-Time) method [TBL92]. In DSD/SST, the

weak form is written not only over the spatial domain, but instead the space-time domain. Thus, we do not need to modify the equations to account for the deforming domain.
In the following, we will define the finite element function spaces for the DSD/SST method. The time interval $(0,T)$ is divided into subintervals $I_n = (t_n, t_{n+1})$, where n defines the time level. We set $\Omega_n = \Omega_{t_n}$ and $\Gamma_n = \Gamma_{t_n}$. Thus, a space-time slab Q_n is defined as the volume enclosed by the two surfaces Ω_n, Ω_{n+1} and the lateral surface P_n. P_n is described by Γ_t as it traverses I_n.
We use first-order interpolation for all degrees of freedom. Thus, a SUPG/PSPG stabilization technique is used in order to fulfill the LBB condition [DH03]. The finite-element interpolation and weighting function spaces for velocity, pressure and temperature for every space-time slab are defined as

$$(\mathcal{S}_{\mathbf{u}}^h)_n = \{\mathbf{u}^h \in [H^{1h}(Q_n)]^{n_{sd}} \mid \mathbf{u}^h \doteq \mathbf{g}^{f,h} \text{ on } (P_n)_{\mathbf{g}}\}, \tag{7.33}$$
$$(\mathcal{V}_{\mathbf{u}}^h)_n = \{\mathbf{w}^h \in [H^{1h}(Q_n)]^{n_{sd}} \mid \mathbf{w}^h \doteq \mathbf{0} \text{ on } (P_n)_{\mathbf{g}}\}, \tag{7.34}$$
$$(\mathcal{S}_p^h)_n = (\mathcal{V}_p^h)_n = \{p^h \in H^{1h}(Q_n)\}, \tag{7.35}$$
$$(\mathcal{S}_T^h)_n = \{T^h \in H^{1h}(Q_n) \mid T^h \doteq g^{T,h} \text{ on } (P_n)_g\}, \tag{7.36}$$
$$(\mathcal{V}_T^h)_n = \{v^h \in H^{1h}(Q_n) \mid v^h \doteq 0 \text{ on } (P_n)_g\}. \tag{7.37}$$

The stabilized space-time formulation for equations (7.20) and (7.21) then reads:
Given $(\mathbf{u}^h)_n^-$, find $\mathbf{u}^h \in (\mathcal{S}_{\mathbf{u}}^h)_n$ and $p^h \in (\mathcal{S}_p^h)_n$ such that:

$$\begin{aligned}
&\int_{Q_n} \mathbf{w}^h \cdot \rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \boldsymbol{u}^h \right) dQ + \int_{Q_n} \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}^h(p^h, \mathbf{u}^h)\, dQ \\
&\quad + \int_{Q_n} q^h \nabla \cdot \mathbf{u}^h \, dQ + \int_{\Omega_n} (\mathbf{w}^h)_n^+ \cdot \rho \left((\mathbf{u}^h)_n^+ - (\mathbf{u}^h)_n^- \right) d\Omega \\
&\quad + \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \tau_{\mathrm{MOM}} \frac{1}{\rho} \left[\rho \left(\frac{\partial \mathbf{w}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \boldsymbol{w}^h \right) + \nabla q^h \right] \\
&\quad \cdot \left[\rho \left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \nabla \boldsymbol{u}^h \right) - \nabla \cdot \boldsymbol{\sigma}^h(p^h, \mathbf{u}^h) \right] dQ \\
&\quad + \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \tau_{\mathrm{CONT}} \nabla \cdot \mathbf{w}^h \rho \nabla \cdot \mathbf{u}^h \, dQ = \int_{P_n} \mathbf{w}^h \cdot \mathbf{h}^{f,h} \, dP,
\end{aligned} \tag{7.38}$$

holds for all $\mathbf{w}^h \in (\mathcal{V}_{\mathbf{u}}^h)_n$ and $q^h \in (\mathcal{V}_p^h)_n$.
The stabilized space-time formulation for the heat equation (7.24) reads:

Given $(T^h)_n^-$, find $T^h \in (\mathscr{S}_T^h)_n$ such that:

$$\begin{aligned}
&\int_{Q_n} v^h \cdot \rho c_p \left(\frac{\partial T^h}{\partial t} + \mathbf{u}^h \cdot \nabla T^h \right) dQ + \int_{Q_n} \nabla v^h \cdot \kappa \nabla T^h \, dQ \\
&\quad - \int_{Q_n} v^h \, \phi \, dQ + \int_{\Omega_n} (v^h)_n^+ \rho c_p \left((T^h)_n^+ - (T^h)_n^- \right) d\Omega \\
&\quad + \sum_{e=1}^{(n_{el})_n} \int_{Q_n^e} \tau_{\text{TEMP}} \frac{1}{\rho c_p} \left[\rho c_p \left(\frac{\partial v^h}{\partial t} + \mathbf{u}^h \cdot \nabla v^h \right) \right] \\
&\quad \cdot \left[\rho c_p \left(\frac{\partial T^h}{\partial t} + \mathbf{u}^h \cdot \nabla T^h \right) - \nabla \cdot \kappa \nabla T^h - \phi \right] dQ \\
&\quad = \int_{P_n} v^h h^{T,h} \, dP,
\end{aligned} \tag{7.39}$$

holds for all $v^h \in (\mathscr{V}_T^h)_n$. The following notation is used:

$$\begin{aligned}
\left(\mathbf{u}^h \right)_n^{\pm} &= \lim_{\zeta \to 0} \mathbf{u}^h (t_n \pm \zeta) \\
\int_{Q_n} \ldots dQ &= \int_{I_n} \int_{\Omega_{t_n}} \ldots d\Omega dt \\
\int_{P_n} \ldots dP &= \int_{I_n} \int_{\Gamma_{t_n}} \ldots d\Gamma dt
\end{aligned} \tag{7.40}$$

The stabilization parameters τ_{MOM}, τ_{CONT} and τ_{TEMP} are based on expressions given in [PB17]. We use a Newton-Raphson method to solve equation (7.38) and (7.39). The flow and temperature fields are coupled strongly using a fixed-point iteration until convergence is reached. The stress contributions in the SUPG/PSPG stabilization terms (fifth term in equation (7.38) and fourth term in equation (7.39)) are zero, since they involve second-order derivatives. We improve the consistency of our method by employing a least-squares recovery technique for these terms [JCWS99]. In order to solve the resulting linear system of equations within each Newton iteration, we use a GMRES solver with an ILUT preconditioner.

7.4. NUMERICAL EXAMPLES

7.4.1. TWO-DIMENSIONAL ISOTHERMAL FLOW

We want to show that simulations based on the new method for grid generation produces the same high-quality results as already established methods like SRMUM or XFEM. Therefore, we simulate the isothermal flow of a plastic melt, described by the Carreau model, in a 2D cross section of a twin-screw extruder. The Carreau parameters are given in Table 7.1. The screw geometry is generated based on an adapted version of Booy's description as presented in [FHM+12, HBE18]. The screw parameters are given in Table 7.2. The rotation speed of the screws is $\omega_s = 60$ rpm in mathematically positive direction. We set the rotational velocity on the screws as a Dirichlet boundary condition and a no-slip condition on the barrel. In [FHM+12], Stokes flow was used. Since we use

Navier-Stokes equations to model the flow, we employ a density of $\rho = 1\ kg/m^3$ in order to make the results comparable. We employ a time step size of $\Delta t = 0.00625s$ or

η_0	1290	$Pa\,s$
η_∞	0	$Pa\,s$
n	0.559	-
λ	0.112	s

Table 7.1: Carreau parameters.

Screw radius R_s	15.275 mm
Center line distance C_l	26.2 mm
Screw-screw clearance δ_s	0.2 mm
Screw-barrel clearance δ_b	0.15 mm

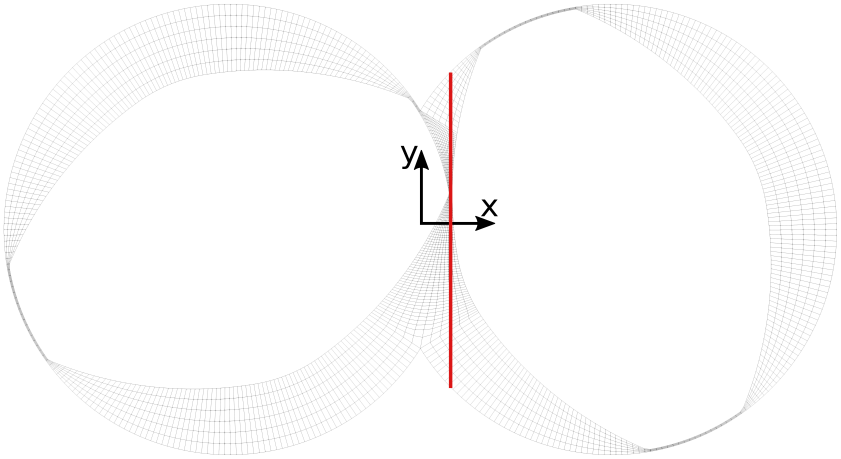
Table 7.2: Screw geometry parameters.

$2.25\,°/s$. The resulting flow can be considered as quasi-steady or instantaneous because the time-scales of the momentum diffusion are very small compared to the process itself. The resulting Reynolds number inside the small gap region is $Re = 0.000003$. Therefore, it is sufficient to compare the solution at individual screw orientations. Two screw orientations, namely $\theta = 0°$ and $\theta = 112.5°$ have been used in [HBE18]. It is demonstrated that $\theta = 112.5°$ is the more complex orientation. Thus, we will only compare results for that orientation. In order to verify the general applicability of the method, we use 4 different mesh resolutions. One advantage of the presented method is that we have to construct the spline parameterization only once and can then extract the scaffolding for all desired mesh resolutions. The region of special interest is the intermeshing area between the two screws. A high pressure drop drives the flow solution resulting in high velocities and high shear rates, see Figure 7.7b. Thus, resolving this part correctly is extremely important. Therefore, we only refine our mesh inside this region and keep the mesh resolution inside the two C-grids constant. The only exception is the first mesh, where we slightly reduce the resolution for the C-grid in order to avoid highly stretched elements. We evaluate the spline-parameterization of the C-grids as well as the separator for all mesh points on the screw, meaning $n_s = n_\mu$. Inside the separator we use a scaffolding with 12 elements in μ-direction. This number is kept constant throughout the refinement of the background mesh. All mesh quantities are given in Table 7.3. In the following, we will refer to them by the total number of elements in screw direction 'n_s total'.

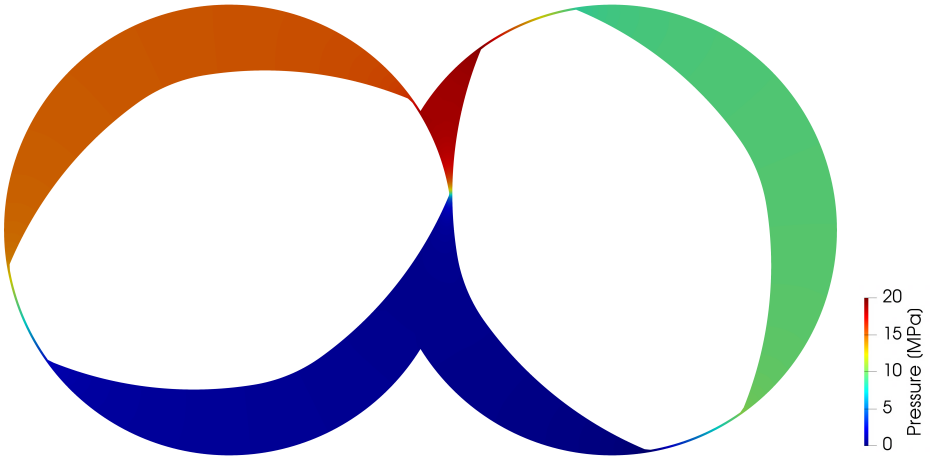
We compare the velocity results inside the intermeshing area along a line in y-direction at $x = 2.077\ mm$. The line is visualized in red for mesh 280 in Figure 7.7a. As a reference solution, we use flow results computed with the same time step size on a SRMUM mesh with 1000 elements in screw and 20 elements in radial direction. The plot over line for the normalized y-velocity y_{vel}/V_p, with $V_p = 2\pi R_s \omega_s$ is shown in Figure 7.8. Looking at the overall plot over line in Figure 7.8a, we can not observe major differences between the

mesh	n_s C-grid	n_s separator	n_s total	n_r	# elements	n_μ	n_v total
1	200	80	280	6	3360	12	280
2	300	160	460	10	9200	12	460
3	300	300	600	12	14400	12	600
4	300	600	900	18	32400	12	900

Table 7.3: Mesh discretization for 2D convergence study.



(a) Mesh and plot over line.



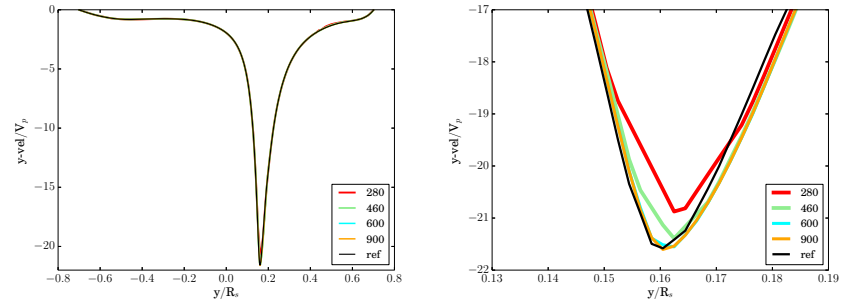
(b) Pressure distribution.

Figure 7.7: Mesh as well as pressure results for the orientation $\theta = 112.5°$.

results computed on different meshes compared to the reference solution. The velocities differ by less than 1 %. However, the peak velocities differ more. Figure 7.8b shows that this difference decreases with increasing mesh resolution. The results are also in accordance with the results presented in [FHM+12].

7.4.2. TWO-DIMENSIONAL TEMPERATURE-DEPENDENT FLOW INSIDE MIXING ELEMENTS

In the previous section, we have shown that the meshes based on the spline-based parameterization technique produce valid results. However, the true advantage over SR-



(a) (b)

Figure 7.8: Velocity plots over line for orientation $\theta = 112.5°$ – (a) velocity profile over entire line (b) close view. The numbers in the legend denote meshes based on n_s total, see Table 7.3. The reference solution ref is computed with the time step based on a SRMUM mesh with 1000 elements in screw and 20 elements in radial direction.

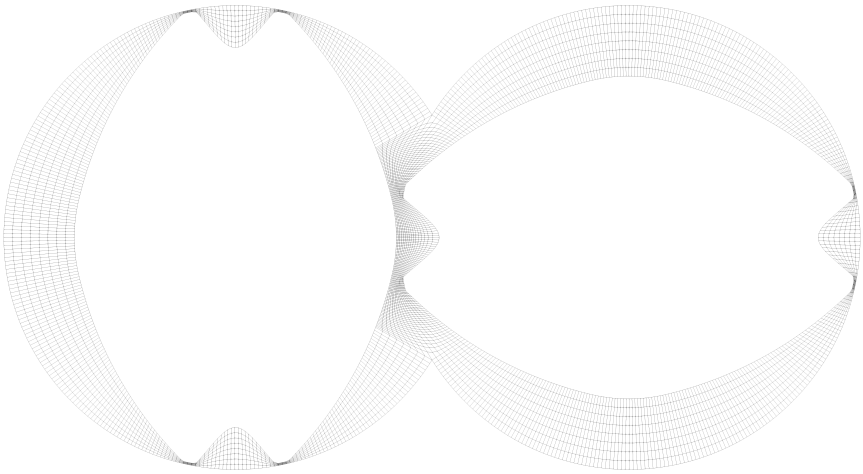
MUM only becomes apparent in the context of non-convex screw shapes - shapes that SRMUM cannot handle at all. We use two different screw configurations that are inspired by the screw design that has already been given in Table 7.2. The resulting screw geometries are given in Figures 7.17 and 7.18.

mesh	n_s C-grid	n_s separator	n_s total	n_r	# elements
1	150	60	210	4	1680
2	300	120	420	8	6720
3	600	240	840	16	26880

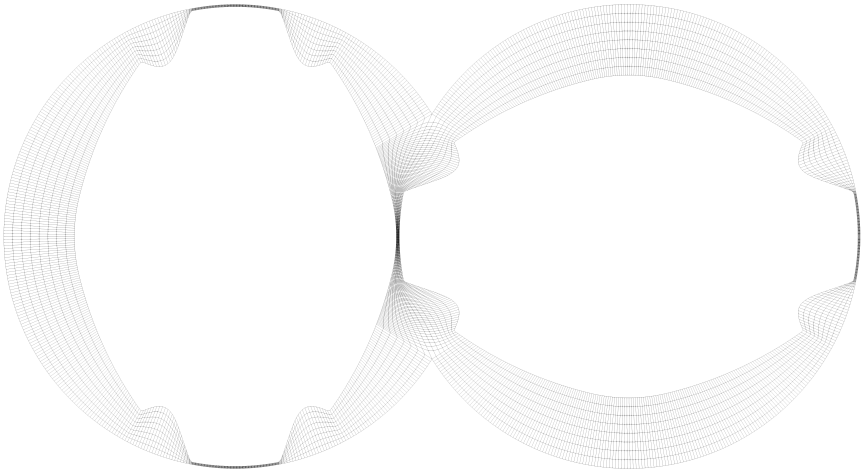
mesh	n_μ	n_v total	$n_{slices}\ \pi$	memory savings
1	12	210	101	22 %
2	12	420	101	61 %
3	12	840	101	81 %

Table 7.4: Mesh discretization for 2D mixing elements for Config. 2.

Config. 2 can be considered slightly more complex since the screw geometry has sharper edges compared to Config. 1. We employ several meshes for Config. 1 that will be later used in order to perform a mesh convergence study. Mesh details are given in Table 7.4 and 7.5, including the memory savings obtained by only storing a scaffolding. Note, that the mesh for Config. 2 is slightly finer than the second mesh of Config. 1. This is due to the more complex shape of the screw profile, which necessitates using more grid points to properly capture the sharp edges.
Similar to the previous section, we evaluate the spline parameterization along with the computed control mapping. Also in this non-convex case, the reparameterization is used to ensure that the C-grid parameterization with transfinite interpolation does not fold. Inside the separator, we again evaluate the spline control mapping composition in a uniform grid comprised of 12 elements in μ-direction and $n_v = n_s$ in v-direction.



(a) Config. 1



(b) Config. 2

Figure 7.9: Two 2D screw configurations for different mixing elements.

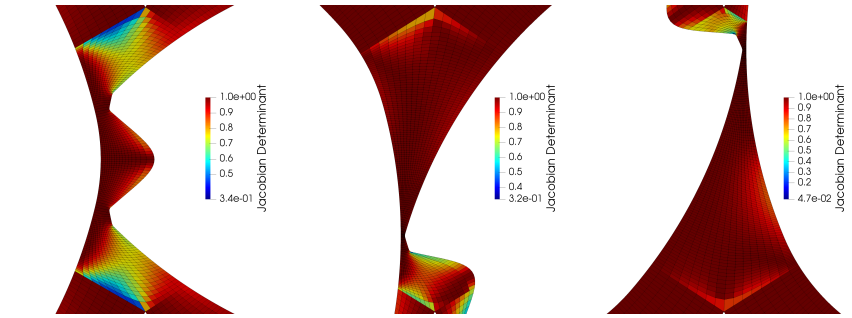
The resulting interpolated fine meshes inside the separator are given for different angles for Config. 1 in Figure 7.10 and Config. 2 in Figure 7.11. In order to evaluate the mesh quality, we use the scaled Jacobian determinant calculated by the *'Mesh quality'* filter in Paraview [Aya15]. We obtain highly distorted elements in particular in the vicinity of concave corners, but the scaled Jacobian determinant is never negative. This is extremely important because the finite-element discretization can cope with highly distorted elements but fails in case of negative determinants.

In the following, we aim to demonstrate the advantage of the presented meshing method.

n_s C-grid	n_s separator	n_s total	n_r	# elements
400	140	540	8	8640

n_μ	n_v total	$n_{slices}\ \pi$	memory savings
12	540	101	63 %

Table 7.5: Mesh discretization for 2D mixing elements for Config. 2.



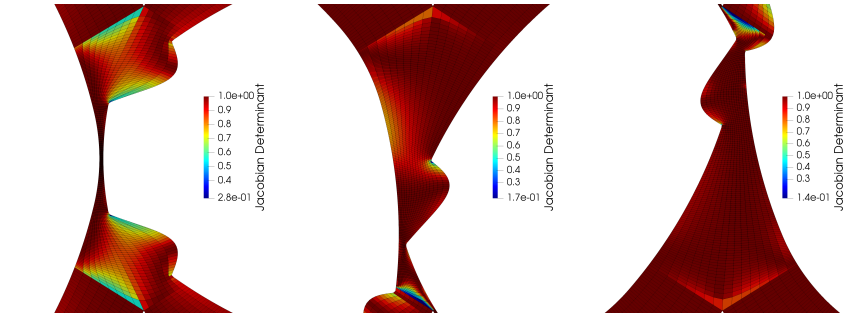
(a) 0.0° (b) 32.4° (c) 129.6°

Figure 7.10: Scaled Jacobian determinant inside the separator or intermeshing area for Config. 1 for different angles on mesh 2.

In case of isothermal flow, it is not crucial to have matching discretization for consecutive screw orientations due to the quasi-steady behavior of the flow. However, taking temperature effects into account results in equations where a solution at the next time step strongly depends on the previous one. Therefore, we simulate a temperature-dependent plastic melt inside the two 2D mixing elements. The melt is modeled using the Cross-WLF model. The parameters are given in Table 7.6. Note, that the parameters have been selected for testing purposes but are in inspired by those of polypropylene. The plastic melt has density $\rho = 700\ kg/m^3$, specific heat $c_p = 2400\ J/(kg\ K)$ and thermal conductivity $\kappa = 10.0\ W/(m\ s)$. The screws rotate in mathematically positive direction with $\omega = 60$ rpm. For the flow, we set a no-slip condition on the barrel and the rotational velocity as Dirichlet condition on the screws.
Concerning the temperature, the screws are considered to be adiabatic. On the barrel we set a Dirichlet temperature condition as $T_{barrel} = 473K + (10x)/0.03K/m$ and by that,

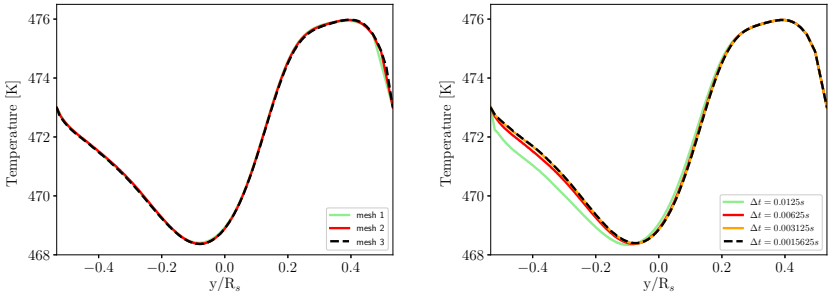
$D1$	5.0e+13	$Pa\ s$
τ^*	2500.0	Pa
n	0.29	-
T_{ref}	263.15	K
$A1$	28.32	-
$A2$	51.60	K

Table 7.6: Cross-WLF parameters.



(a) 0.0° (b) 32.4° (c) 129.6°

Figure 7.11: Scaled Jacobian determinant inside the separator or intermeshing area for Config. 2 for different angles.



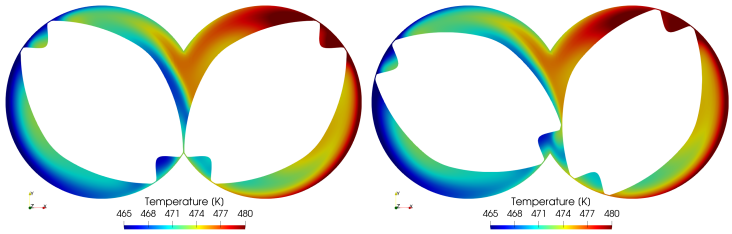
(a) Refinement in space with $\Delta t = 0.0625s$. (b) Refinement in time on mesh 2.

Figure 7.12: Mesh and time step refinement study for Config. 1 without viscous dissipation. The temperature results are compared along a line from the lower to upper cusp point at time $t = 1.625s$.

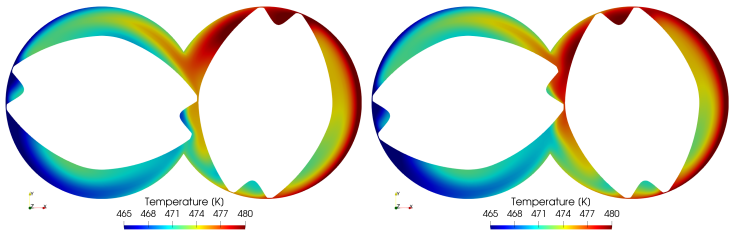
generate a linear increase of the temperature from the left to the right barrel. The initial condition is $T_0 = 473K$. It is important to note that this constitutes a test case that has no connection to a real twin-screw extruder application. It merely demonstrates the validity of the method, as cold/hot melt is constantly pushed from left to right and vice versa. We use different time step sizes to show that the solutions are independent of the time step. Thus, we have to interpolate linearly between slices for most of the resulting screw orientation. Assuming that each individual slice is bijective, a linear interpolation in z-direction will lead to a bijective volumetric mesh.

CONFIG. 1:

We investigate the temperature field for Config. 1 for different time steps on different meshes. In a first step we neglect the effects of viscous dissipation to solely demonstrate how the hot and cold melt is pushed from left to right and vice versa. We use three different meshes to show that the results are mesh independent. Therefore, we



(a) $t = 1.625s$ (b) $t = 1.675s$

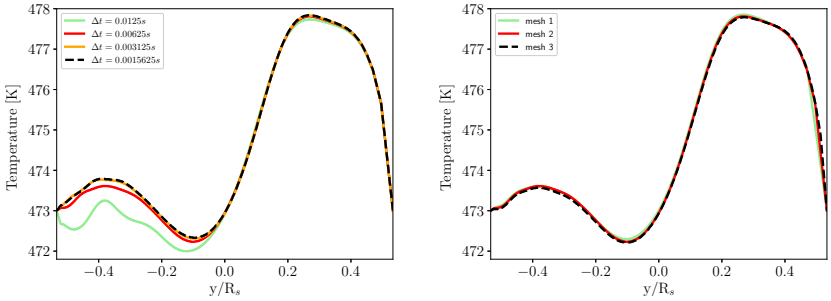


(c) $t = 1.725s$ (d) $t = 1.775s$

Figure 7.13: Temperature field for Config. 1 for selected time steps demonstrating relevant physical effects without viscous dissipation. The results have been computed on mesh 2 using a time step of $\Delta t = 0.003125s$.

compare the temperature results along a line between the lower and upper cusp point at time $t = 1.625s$. The time step size is $\Delta t = 0.00625s$. The results are given in Fig. 7.12a. Mesh convergence is clearly visible. The results on the coarsest mesh already show very good agreement to the ones obtained on the finest mesh. In the following, we investigate the choice of an appropriate time step size. We compute the temperature results on mesh 2 for four different time steps, 7.12b. We can observe a difference between results for the different time step sizes especially inside the small gap region. However, even for $\Delta t = 0.0125s$ we obtain results with a reasonable error for industrial applications. The difference between results obtained with $\Delta t = 0.0625s$ and the finest time step size $\Delta t = 0.0015625s$ are less than 1 %.

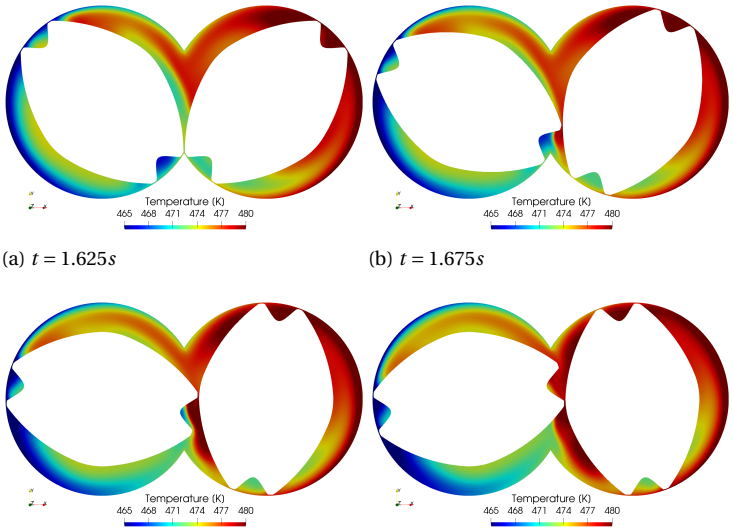
Figure 7.13 shows the temperature field at different angles computed on mesh 2 with $\Delta t = 0.003125s$. As already described, cold melt is transported by the screws from left to right. Inside the recessed portion of the screw in the lower part of the left barrel, cold melt is transported into the intermeshing area, see Figure 7.13a. At time $t = 1.675s$, the cold fluid has been transported into the intermeshing area and starts to be convected into the lower part of the right barrel. The flow direction inside the small gaps between the screw is negative, pushing warmer melt into this area. In the following time steps, $t = 1.725s$ and $t = 1.775s$, more and more warm melt is pushed through the small gap into the lower part of the intermeshing area, leading to a temperature increase there. The temperature field emerges smoothly in time showing the good quality of the underlying meshes as well as numerical methods. In the following, we include viscous dissipation. Thus, we expect the melt to heat up due to high shear rates inside the small gaps. Again, we analyze the effect of the mesh and time step size. The results are shown in Fig. 7.14.



(a) Refinement in time on mesh 2.

(b) Refinement in space with $\Delta t = 0.0625s$.

Figure 7.14: Mesh and time step refinement study for Config. 1 including viscous dissipation. The temperature results are compared along a line from the lower to upper cusp point at time $t = 1.625s$.



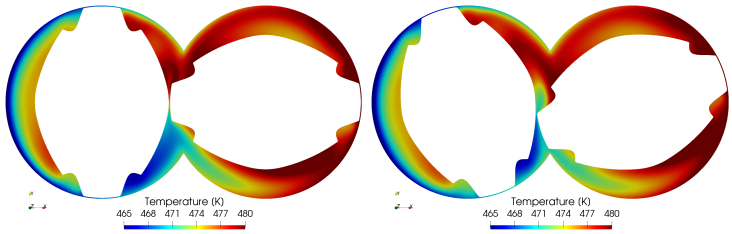
(a) $t = 1.625s$

(b) $t = 1.675s$

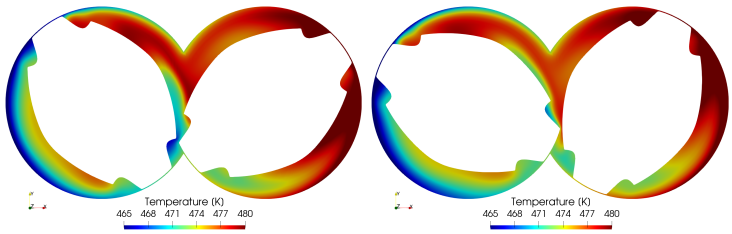
(c) $t = 1.725s$

(d) $t = 1.775s$

Figure 7.15: Temperature field for Config. 1 for selected time steps demonstrating relevant physical effects including viscous dissipation. The results have been computed on mesh 2 using a time step of $\Delta t = 0.003125$ s.



(a) $t = 1.5s$ (b) $t = 1.55s$



(c) $t = 1.6s$ (d) $t = 1.675s$

Figure 7.16: Temperature field for Config. 2 for selected time steps demonstrating relevant physical effects including viscous dissipation.

Similar to the case without viscous dissipation, we see a good convergence in time and space. However, the difference of the solution between the coarsest and finest time step increases especially inside the small gap region. This shows the additional complexity introduced to the simulation by viscous dissipation. The temperature field at different angles computed on mesh 2 with time step $\Delta t = 0.003125s$ is shown in Figure 7.15. Comparing the results with and without viscous dissipation, we can clearly observe the effects of the heating of the melt due to viscous dissipation. At time $t = 1.675s$, heated melt inside the small gap is pushed into the lower part of the intermeshing area. This continues for the following time steps, leading to a major increase of temperature in that area. The temperatures are more than four degrees above those resulting from neglecting viscous dissipation. However, the convective behavior of the melt is similar to the case with no viscous dissipation.

CONFIG. 2:

For Config. 2, we analyze the temperature and flow behavior at an earlier stage, see Figure 7.16. Based on the numerical experiments for Config. 1, we use a time step of $\Delta t = 0.003125s$. In this configuration, the cold melt is not transported inside the recessed portion, but is pushed into the intermeshing area due to a larger gap region between screw and barrel, see Figure 7.16a. Between time $t = 1.55$ s and $t = 1.6s$, the melt is pushed upwards out of the cold temperature part, which decreases the temperature in the upper part between the screws. At time $t = 1.675s$ we can observe an increase of temperature in the lower part of the intermeshing region. The direction of flow inside the screw-screw gap has changed sign and the melt is pushed downwards through the

gap. Again, the melt is heated up especially inside this small gap due to high shear rates. All these observations are in line with what one would expect by purely looking at flow results.
The results for the two configurations show the potential of the presented spline-based meshing approach. We are able to generate high-quality meshes for extremely challenging moving domains. Only generating a certain amount of slices and interpolating all other meshes seems to be a valid approach. Due to the space-time approach of the method, this unsteady 2D test case is already a proof of concept for 3D. Furthermore, it is noteworthy that the time spent for updating the mesh was less than 0.1 %. The simulations have been run on 24 cores on the Intel Xeon based RWTH cluster using an MPI parallelization. For example, computing one revolution for Config. 1 on mesh 2 using a time step of $\Delta t = 0.00625s$ takes $115s$. The time spent for the mesh update was only $0.02s$.

Remark. The appeal of the presented methodology becomes apparent when performing mesh refinement studies as well as time step size studies. The spline parameterization for the screw only has to be generated once at the beginning. For a mesh refinement study only the scaffolding needs to be regenerated which is a simple evaluation of the spline parameterization. A time step refinement study is even simpler. Given a scaffolding for instances of Θ in the interval $[0,\pi]$, we can simply compute the mesh at any time instance by interpolating between the generated instances. Thus, it is not necessary to generate any new mesh in case one aims to adapt the time step size.

7.4.3. THREE-DIMENSIONAL APPLICATION CASE

Within this section, we aim to show the functioning of the spline-based parameterization technique for real 3D applications. Once again, we consider the temperature-dependent flow of a plastic melt through a complex screw geometry. The plastic melt is modeled using the Cross-WLF model. The model parameters employed are based on a polypropylene from the product portfolio of a leading raw material manufacturer. The parameters are given in Table 7.8. The plastic melt has density $\rho = 710\ kg/m^3$, specific heat $c_p = 2400\ J/(kg\ K)$ and thermal conductivity $\kappa_0 = 0.5\ W/(m\ s)$.

Screw radius R_s	0.156 m
Center line distance C_l	0.262 m
Screw-screw clearance δ_s	0.004 m
Screw-barrel clearance δ_b	0.004 m
Pitch length p_l	0.28 m

Table 7.7: Geometry parameters of a 3D mixing screw element for temperature-dependent flow.

$D1$	1.2e+14	$Pa\ s$
τ^*	25680.0	Pa
n	0.29	-
T_{ref}	263.15	K
$A1$	28.32	-
$A2$	51.60	K

Table 7.8: Cross-WLF parameters.

The 2D screw geometry cross section is a simplified combination of the two 2D configurations of the previous section, see Figure 7.17. The screw geometry parameters are given in Table 7.7. The 3D setup is shown in Figure 7.18. For simplicity we only consider a single screw element. The screws rotate in mathematically positive direction with $\omega = 120$ rpm. The computational domain is extended at the inflow and outflow. By that, a solely

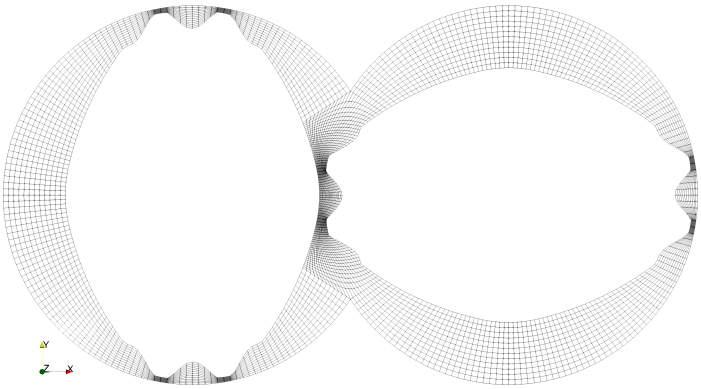


Figure 7.17: 2D cross section.

n_s C-grid	n_s separator	n_s total	n_r	n_a	# elements	n_μ	n_v total	n_{slices} π
150	70	220	12	300	1584000	220	16	101

Table 7.9: Mesh discretization for 3D mixing elements.

positive velocity in z-direction is guaranteed at the outflow. This is important since we set a natural boundary condition and thus, negative velocities might cause instabilities. Furthermore, this setting is also similar to a industry-like twin-screw extruder where the screw section leads to the die. The extended inflow circumvents high shear rates in the inflow area, which would result in an unnaturally high temperature increase. The extension is achieved by relaxing the 2D screw surface over a distance of 0.28 m to a circle with a radius of 0.06 m. For the mesh, we simply interpolate between the original 2D mesh and a structured mesh between the two circles. We use a background mesh with $n_s = 220$ elements on the screw, split into 70 elements inside the separator and 150 for the C-grid. $n_r = 12$ elements are used in radial direction. A full pitch length is discretized using 200 elements in axial direction and the inflow and outflow extensions are discretized with 50 elements each. The scaffolding generated by the spline parameterization consists of all points in circumferential direction, $n_s = n_v$ and $n_\mu = 16$ inside the separator. In order to account for the rotation, the scaffolding is evaluated for 101 equally distributed instances of Θ in the interval $[0, \pi]$. Storing only the scaffolding instead of the full mesh for each instance results in memory savings for the mesh of 75 %. All mesh parameters are listed in Table 7.9. We simulate a scenario with a mass flow rate of $\dot{m} = 25560\ kg/h$. This is achieved by setting a uniform inflow velocity at the inlet. The barrel is heated with $T_{barrel} = 473K + 10K * z/1.12m$ and the screws are considered to be adiabatic. The inflow temperature is $T_{inflow} = 473\ K$. In order to obtain a good initial condition for the temperature, we compute a steady solution where we neglect the viscous dissipation term and increase the conductivity by a factor of 10. The time step size is $\Delta t = 0.005s$ which is chosen based on the results of the previous section. The results have been computed on 360 cores on the Intel Xeon based Juelich cluster. Computing one revolution

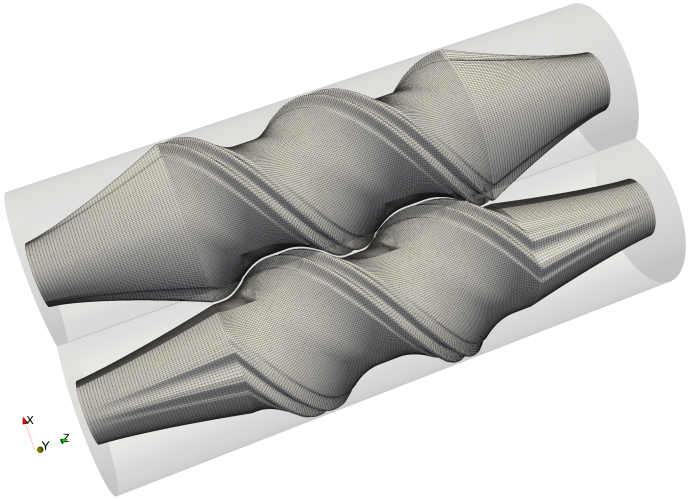
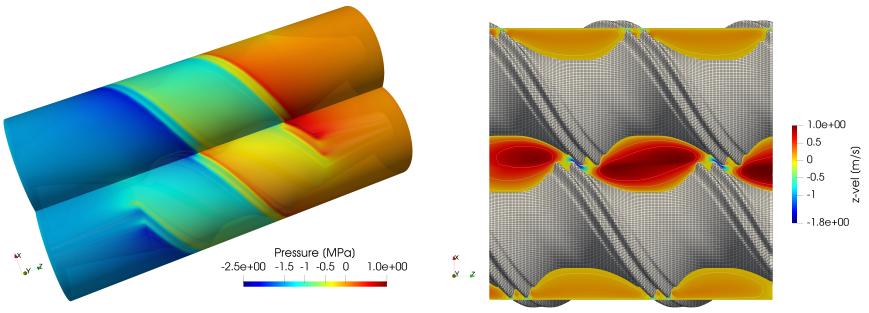


Figure 7.18: 3D sketch of extruder including in/outflow extension.

takes 5832s, whereas the part spent for the mesh update is only 0.77s which shows the efficiency of the presented meshing approach. The flow results are presented in Figure 7.19. Operating with the chosen design results in a pressure build-up over the screw element, see Figure 7.19b, which could be used to transport the fluid into, for instance, an injection mold. The velocity in z-direction is given in a plane at $y = 0.074\ m$ in Figure 7.19a. A strong backflow behavior can be observed in the small gaps between the screws. This is desirable for twin-screw extruders since it increases their mixing capabilities. Inside the extruder we observe Reynolds numbers up to $Re = 0.1$, which justifies using Navier-Stokes equations. The temperature field reaches a periodic state after roughly 10 revolutions. The temperature results for half a rotation are shown in Figure 7.20. The melt is heated inside the high shear regions in the small gaps between the screws. Furthermore, we can observe how cold melt is pushed forward inside the high velocity regions, especially inside the wide parts of the intermeshing region.

7.5. CHAPTER CONCLUSIONS

Within this work, we presented an efficient and robust meshing strategy that allows to use boundary-conforming finite element methods to compute the unsteady flow of plastic melt inside co-rotating twin-screw extruders. It is suitable for arbitrarily-shaped screw geometries. The method is a combination of spline-based meshing techniques based on EGG and SRMUM. 2D spline-based geometry descriptions are generated for a certain number of screw orientations. The spline description is evaluated at discrete points to obtain a point cloud that is used as a *scaffolding* to adapt a structured background mesh to the actual screw configuration. In contrast to algebraic grid generation, the advantage of the proposed approach is two-fold: on the one hand, it allows for fine tuning of the mesh properties by employing a control mapping (see Section 7.2.4). On



(a) Pressure results for the entire domain.

(b) Contour for the velocity in z-direction at $y = 0.074\ m$.

Figure 7.19: Flow results for $t = 5.0s$.

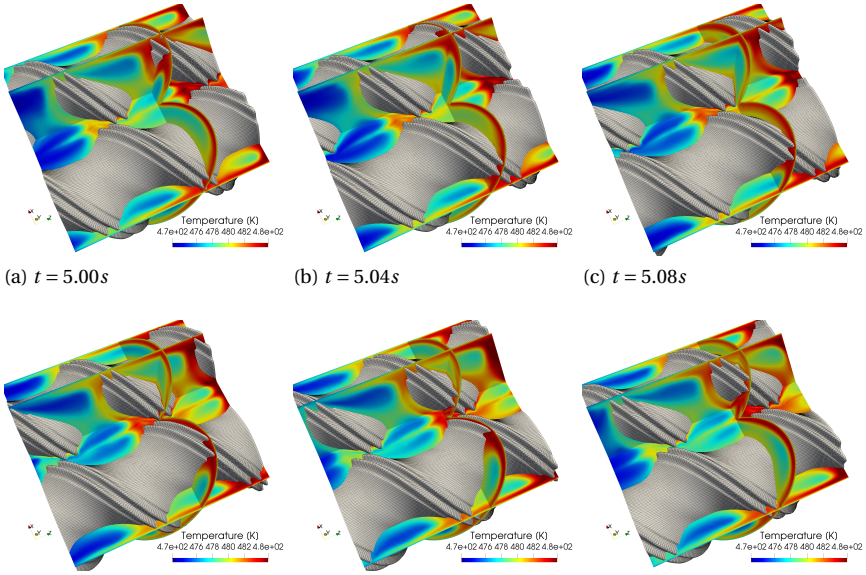
the other hand, refinement studies are easily accomplished by evaluating the composite spline-mapping in a increasing number of points. Additionally, storing only a limited number of points instead of a full 3D mesh rotation saves a lot of memory. The actual mesh update at run-time is very cheap and requires only a fraction of the time spent for the actual solve of the flow solution.

A finite element method based on the Deformable-Spatial-Domain/Stabilized Space-Time (DSD/SST) finite element formulation was used. A 2D test case simulating the flow of an isothermal polymer in a twin-screw extruder cross section served as a validation case for the presented framework including the new meshing strategy. Convergence of the solution has been demonstrated as well as accordance to results from literature. Furthermore, two complex, mixing-element-like, screw shapes have been used to show the robustness of the meshing technique in 2D. We computed unsteady temperature-dependent flow of the plastic melt inside the aforementioned screw shapes. The use of a space-time finite element method already proofs, that the presented method is capable of computing unsteady flow results using boundary-conforming meshes in 3D. We additionally computed the flow of a temperature-dependent plastic melt in 3D for a complex screw shape in a single element twin-screw extruder to further demonstrate the great potential of the presented approach.

In the future, we aim to compute flow results for a larger variety of screw designs. This will be used in order to compare the quantities like residence time distributions or mixing behavior. The advantages of the method already shown for unsteady temperature results can also be exploited by computing solutions of advection-diffusion equations, in order to characterize mixing behavior of the individual screw design.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the research funding which was partly provided by the MOTOR project that has received funding from the European Unions Horizon 2020 research and innovation program under grant agreement No 678727. The computations were conducted on computing clusters supplied by the Juelich Aachen Research Alliance



(a) $t = 5.00s$ (b) $t = 5.04s$ (c) $t = 5.08s$

(d) $t = 5.12s$ (e) $t = 5.16s$ (f) $t = 5.20s$

Figure 7.20: Temperature distribution in mixing element displayed on three planes: xy-plane at $y = 0.074\ m$; a plane with normal $\mathbf{n} = (1.0, -1.5, 0.0)$ and point $\mathbf{r}$ on the plane with $\mathbf{r} = (-0.07, 0.0, 0.0)$; a plane with $\mathbf{n} = (0.0, -1.0, 1.0)$ and $\mathbf{r} = (0, 0, 0.56)$.

7

(JARA) and the RWTH IT Center.

REFERENCES

[ALA04] Bernard Alsteens, Vincent Legat, and Th Avalosse. Parametric study of the mixing efficiency in a kneading block section of a twin-screw extruder. *International Polymer Processing*, 19(3):207–217, 2004.

[Aya15] Utkarsh Ayachit. *The ParaView Guide: A Parallel Visualization Application*. Kitware, Inc., USA, 2015.

[Aza09] Boris Nikolaevich Azarenok. Generation of structured difference grids in two-dimensional nonconvex domains using mappings. *Computational Mathematics and Mathematical Physics*, 49(5):797–809, 2009.

[BdFS] A Buffa, C de Falco, and G Sangalli. Isogeometric analysis: new stable elements for the stokes equation, 2010. to appear on inter. *J. Numer. Meth. Fluids. DOI*, 10.

[BHW00] VL Bravo, AN Hrymak, and JD Wright. Numerical simulation of pressure and velocity profiles in kneading elements of a co-rotating twin screw extruder. *Polymer Engineering & Science*, 40(2):525–541, 2000.

[BS13] Andrea Bressan and Giancarlo Sangalli. Isogeometric discretizations of the stokes problem: stability analysis by the macroelement technique. *IMA Journal of Numerical Analysis*, 33(2):629–651, 2013.

[CDK79] PJ Carreau and D De Kee. Review of some useful rheological equations. *The Canadian Journal of Chemical Engineering*, 57(1):3–15, 1979.

[CW91] Z Chen and JL White. Dimensionless non-newtonian isothermal simulation and scale-up considerations for modular intermeshing corotating twin screw extruders. *International Polymer Processing*, 6(4):304–310, 1991.

[DDMN+14] Audrey Durin, Pierre De Micheli, H-C Nguyen, Chantal David, Rudy Valette, and Bruno Vergnes. Comparison between 1d and 3d approaches for twin-screw extrusion simulation. *International Polymer Processing*, 29(5):641–648, 2014.

[DH03] Jean Donea and Antonio Huerta. *Finite element methods for flow problems*. John Wiley & Sons, 2003.

[FHM+12] Arash Sarhangi Fard, MA Hulsen, HEH Meijer, NMH Famili, and PD Anderson. Adaptive non-conformal mesh refinement and extended finite element method for viscous flow inside complex moving geometries. *International Journal for Numerical Methods in Fluids*, 68(8):1031–1052, 2012.

[GH73] William J Gordon and Charles A Hall. Transfinite element methods: blending-function interpolation over arbitrary curved element domains. *Numerische Mathematik*, 21(2):109–129, 1973.

[HBE18] Jan Helmig, Marek Behr, and Stefanie Elgeti. Boundary-conforming finite element methods for twin-screw extruders: Unsteady-temperature-dependent-non-newtonian simulations. *arXiv preprint arXiv:1901.00725*, 2018.

[HCB05] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *CMAME*, 194:4135–4195, 2005.

[HHME20] Jochen Hinz, Jan Helmig, Matthias Möller, and Stefanie Elgeti. Boundary-conforming finite element methods for twin-screw extruders using spline-based parameterization techniques. *Computer Methods in Applied Mechanics and Engineering*, 361:112740, 2020.

[HI13] J-F Hétu and F Ilinca. Immersed boundary finite elements for 3d flow simulations in twin-screw extruders. *Computers & Fluids*, 87:2–11, 2013.

[HMV18a] J Hinz, M Möller, and C Vuik. Elliptic grid generation techniques in the framework of isogeometric analysis applications. *Computer Aided Geometric Design*, 2018.

[HMV18b] Jochen Hinz, Matthias Möller, and Cornelis Vuik. Spline-based parameterization techniques for twin-screw machine geometries. In *IOP Conference Series: Materials Science and Engineering*, volume 425, page 012030. IOP Publishing, 2018.

[HN81] Cyril W Hirt and Billy D Nichols. Volume of fluid (vof) method for the dynamics of free boundaries. *Journal of computational physics*, 39(1):201–225, 1981.

[IKF01] Takeshi Ishikawa, Shin-ichi Kihara, and Kazumori Funatsu. 3-d non-isothermal flow field analysis and mixing performance evaluation of kneading blocks in a co-rotating twin srew extruder. *Polymer Engineering & Science*, 41(5):840–849, 2001.

[JCWS99] Kenneth E Jansen, S Scott Collis, Christian Whiting, and Farzin Shakib. A better consistency for low-order stabilized finite element methods. *Computer methods in applied mechanics and engineering*, 174(1-2):153–170, 1999.

[Joh05] Richard W Johnson. Higher order B-spline collocation at the greville abscissae. *Applied Numerical Mathematics*, 52(1):63–75, 2005.

[KM07] DM Kalyon and M Malik. An integrated approach for numerical analysis of coupled flow and heat transfer in co-rotating twin screw extruders. *International Polymer Processing*, 22(3):293–302, 2007.

[Kra88] Dieter Kraft. A software package for sequential quadratic programming. *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.

[MH18] Matthias Möller and Jochen Hinz. Isogeometric analysis framework for the numerical simulation of rotary screw machines. I. General concept and early applications. In *IOP Conference Series: Materials Science and Engineering*, volume 425, page 012032. IOP Publishing, 2018.

[MKGJ14] M Malik, DM Kalyon, and JC Golba Jr. Simulation of co-rotating twin screw extrusion process subject to pressure-dependent wall slip at barrel and screw surfaces: 3d fem analysis for combinations of forward-and reverse-conveying screw elements. *International Polymer Processing*, 29(1):51–62, 2014.

[MTH+14] Otto Mierka, T Theis, T Herken, S Turek, V Schöppner, and F Platte. *Mesh Deformation Based Finite Element-Fictitious Boundary Method (FEM-FBM) for the Simulation of Twin-screw Extruders*. Citeseer, 2014.

[OS88] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. *Journal of computational physics*, 79(1):12–49, 1988.

[PB17] Lutz Pauli and Marek Behr. On stabilized space-time fem for anisotropic meshes: Incompressible navier–stokes equations and applications to blood flow in medical devices. *International Journal for Numerical Methods in Fluids*, 85(3):189–209, 2017.

[PT12] Les Piegl and Wayne Tiller. *The NURBS book*. Springer Science & Business Media, 2012.

[RC18] Martin Robinson and Paul W Cleary. Effect of geometry and fill level on the transport and mixing behaviour of a co-rotating twin screw extruder. *Computational Particle Mechanics*, pages 1–21, 2018.

[RO14] Natalie Rudolph and Tim A Osswald. *Polymer rheology: fundamentals and applications*. Carl Hanser Verlag GmbH Co KG, 2014.

[TBL92] T. E. Tezduyar, M. Behr, and J. Liou. A new strategy for finite element computations involving moving boundaries and interfaces–the deforming-spatial-domain/space-time procedure: I. the concept and the preliminary numerical tests. 94(3):339 – 351, 1992.

[VCDV09] Rudy Valette, Thierry Coupez, Chantal David, and Bruno Vergnes. A direct 3d numerical simulation code for extrusion and mixing processes. *International Polymer Processing*, 24(2):141–147, 2009.

[VVD98] Bruno Vergnes, G Della Valle, and Laurent Delamare. A global computer software for polymer flows in corotating twin screw extruders. *Polymer Engineering & Science*, 38(11):1781–1792, 1998.

[WPE+18] P Wittek, GG Pereira, MA Emin, V Lemiale, and PW Cleary. Accuracy analysis of sph for flow in a model extruder with a kneading element. *Chemical Engineering Science*, 2018.

[ZFCH09] Xian-Ming Zhang, Lian-Fang Feng, Wen-Xing Chen, and Guo-Hua Hu. Numerical simulation and experimental validation of mixing performance of kneading discs in a twin screw extruder. *Polymer Engineering & Science*, 49(9):1772–1783, 2009.

8

CONCLUSION

In this chapter we recapitulate the main findings of this work and suggest possible topics for future research.

8.1. GENERAL CONCLUSIONS

This work extends the available NURBS-based geometry parameterization arsenal by techniques that employ the principles of FEA / IGA to a PDE-based problem formulation for the surface to volume problem $\partial\Omega \longrightarrow \Omega$. The techniques are based on the three basic principles from Section 1.3, which we restate here for convenience:

A. Robustness is more important than efficiency;

B. Differentiability counts;

C. Whenever possible, topology changes should be avoided.

In the following, we discuss the manifestations of Points A to C in the techniques developed by this work as well as the key advantages we see in employing a PDE-based problem formulation for the geometrical aspects of an IGA-based CSE workflow.

- As a main manifestation of Point A, the provided techniques approximate a geometry parameterization whose inverse is comprised of harmonic functions in Ω. Hereby, the theoretically predicted bijectivity property enables removing mapping degeneracies through (local or structured) refinement. This possibility is first addressed in Section 2.8 and further developed in Chapter 4, where THB-splines are combined with local refinement based on duality considerations. The algorithmically convenient possibility of trading computational costs in exchange for robustness is a key aspect of the techniques presented in this work. Besides being the main topic of Chapter 4, a posteriori refinement is extensively employed to improve the computational robustness of the twin-screw machine parameterization frameworks developed in Chapters 6 and 7. It plays a key role in Chapter 5

where it enables adopting the so-called *variable basis approach* which performs repeated THB-refinement during each shape optimization iteration and enables tuning the IGA basis to the currents needs. Overall, the improved robustness resulting from (local) refinement is a main contributor for the development of the autonomously operating techniques from Chapters 5 to 7.

- The monolithic treatment of geometry and simulation, made possible by the PDE-based problem formulation, streamlines the computational workflow and simplifies the codebase. This is mainly owed to employing a large subset of the widely-used (and hence readily-available) FEA / IGA techniques for the geometrical aspects of the numerical simulation pipeline, such as dual weighted residual (cf. Section 4.3), (Schur complement) Newton-Krylov (cf. Sections 3.3.1, 3.3.2, 4.2.1 and 5.5.4), mixed finite element techniques (Chapter 3), THB-splines (Chapter 4) and multigrid techniques (Chapters 2 and Chapter 6).

- The methods developed by this work largely conform with Point B. Hereby, differentiability (with respect to the boundary correspondence $\mathbf{x}_D : \partial\hat{\Omega} \rightarrow \partial\Omega$) is greatly facilitated by the PDE-based problem formulation. As a main use case, Chapter 5 proposes a shape optimization algorithm in which the gradient of the objective function, which exhibits a complex dependency on the continuously changing geometry, is assembled from an expression that has been derived fully symbolically. This is accomplished by adding the governing PDE-equations corresponding to both geometry and state variable to the problem formulation in the form of additional equality constraints. The chapter derives expressions for the constituents of the gradient from the combined IGA-residual by making use of the implicit function theorem. It then computationally efficiently assembles the gradient using an adjoint method, completely bypassing the need to fall back on finite-differencing or automatic differentiation. Once an objective function evaluation is completed, the additional costs associated with computing the gradient are low. Hence, algorithms that employ Hessian update strategies, which typically compute the objective function and gradient in tandem, are particularly well-suited.

- Differentiability is a key aspect of the twin-screw machine geometry parameterization frameworks from Chapters 6 and 7. The chapters ensure that the Dirichlet data $\partial\Omega$ is ($C^{\geq 1}$)-continuous in the rotational angle θ by blending a fixed number of uniformly-spaced reparameterization functions over the entire θ-interval (see Sections 6.8 and 7.2.4). As a result, the PDE-solution becomes a differential function of θ. This enables an approach that hierarchically fills an initially empty database with planar parameterizations for a large number of discrete angles θ_i. The continuous dependency allows for interpolation within the partially-filled database, which, in turn, reduces computational costs. It furthermore enables generating volumetric parameterizations by taking a large number of planar parameterizatons from the database and stacking them in the z-direction. This process essentially collocates a (in z-direction) diffeomorphic volumetric parameterization by sweeping a family of planar parameterizations in z-direction.

- Chapters 6 and 7 avoid topology changes by employing a parametric domain com-

prised of three macro elements (patches) in combination with a sliding grid approach. Along with the θ-continuous Dirichlet data, this is a main contributor to the database-driven approach developed in the chapters. Furthermore, in combination with differentiability (Point B) it constitutes a key ingredient for generating volumetric parameterizations, which sweep the three families of planar parameterizations (the two C-type grids and the separator) over the full θ interval. Hereby, the number of patches is fixed but the number of elements per patch may vary and are tuned by the knot vector which is adaptively reselected based upon the current needs.

8.2. OUTLOOK

In the following, we conceptualize several topics for future research and present first results.

8.2.1. MULTIPATCH DOMAIN OPTIMIZATION

The concept of domain optimization introduced in Section 4.4, which enables controlling the parametric properties of the PDE-solution, is not limited to singlepatch domains. The multipatch approach from Section 3.3.2 assumes that the mutually disjoint $\hat{\Omega}_i$ which cover the parametric domain

$$\hat{\Omega} = \mathrm{Int}\left(\bigcup_{i=1}^{n} \overline{\hat{\Omega}}_i\right) \tag{8.1}$$

are transformations of the unit quadrilateral $\tilde{\Omega} = (0,1)^2 \subset \mathbb{R}^2$ under a set of affine maps $\hat{\mathbf{m}}_i : \tilde{\Omega} \to \hat{\Omega}_i$. Lifting this restriction enables choosing a set $\mathbf{m}_i : \hat{\Omega}_i \to \hat{\Omega}$, $i \in \{1,\dots,n\}$ such that the parametric properties of the resulting patchwise composite mappings $\mathbf{x}_h^i : \tilde{\Omega} \to \Omega$, with $\mathbf{x}_h^i(\boldsymbol{\mu}) \simeq \tilde{\mathbf{x}}|_{\hat{\Omega}_i} \circ \mathbf{m}_i \circ \hat{\mathbf{m}}_i(\boldsymbol{\mu})$ (cf. (3.32)) are tuned as desired. This induces a global control mapping $\mathbf{s} : \hat{\Omega} \to \hat{\Omega}$, with $\mathbf{s}|_{\hat{\Omega}_i} = \mathbf{m}_i$. A reparameterization is conveniently achieved by differentiating with respect to the components of $\mathbf{s}(\boldsymbol{\xi})$ instead of $\boldsymbol{\xi}$, in the residual from (3.9). Since $\mathbf{s} : \hat{\Omega} \to \hat{\Omega}$ maps the convex parametric domain onto itself, methods that select a suitable reparameterization may exploit the maximum principle, as in Section 4.4.1. We seek a reparameterization as the unique minimizer of the following convex optimization problem:

$$\frac{1}{2}\sum_{i=1}^{n}\int_{\tilde{\Omega}} \left\| \left[\frac{\partial \mathbf{s}|_{\hat{\Omega}_i}}{\partial \hat{\mathbf{m}}_i}\right]\left[\frac{\partial \hat{\mathbf{m}}_i}{\partial \boldsymbol{\mu}}\right] \right\|^2 \mathrm{d}\boldsymbol{\mu} \to \min_{\mathbf{s}\in\mathcal{V}_h^2}, \quad \text{s.t.} \quad \mathbf{s}(\boldsymbol{\xi}) = \boldsymbol{\xi} \quad \text{on} \quad \partial\hat{\Omega}, \tag{8.2}$$

where $\mathcal{V}_h \subset H^1(\hat{\Omega})$ as before. Here, we have made use of the chain rule for an expression of the differential of $\mathbf{s}$ with respect to the reference coordinate system in $\tilde{\Omega}$. Figures 8.1 and 8.2 show the bat-shaped geometry from Section 3.4.3 before and after reparameterization, along with the corresponding parametric domains.

Figure 8.2 reveals that reparameterization with (8.2) has a regularizing effect on the parametric properties of the mapping. In particular, the reparameterization *weakly* enforces higher-order interface continuity (sufficiently distant from the extraordinary vertex) without the need to manually couple DOFs. However, the reparameterization increases the

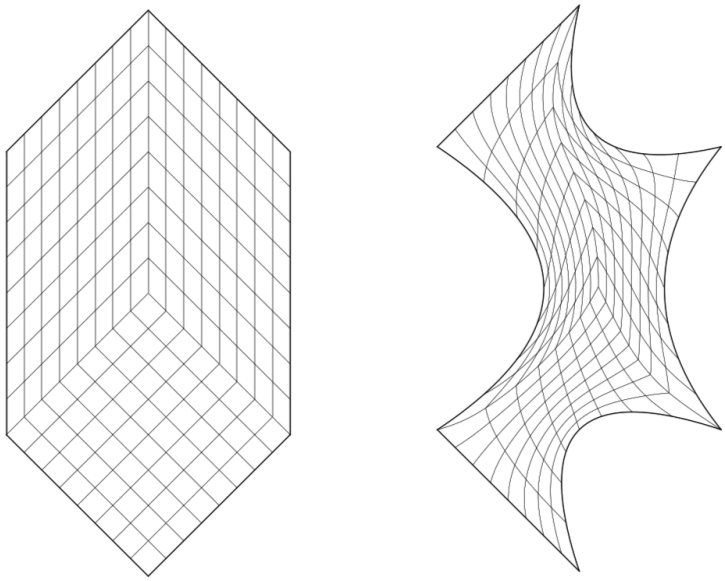


Figure 8.1: The bat-shaped geometry parameterization in the absence of reparameterization.

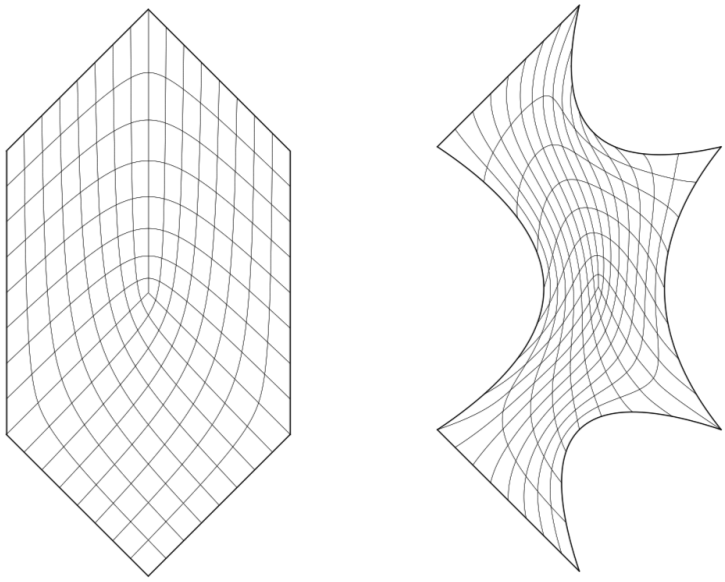


Figure 8.2: The bat-shaped geometry upon reparameterization of $\hat{\Omega}$.

variance in the size of elements. As in Section 4.4.1, the isoline distribution can be tuned by extending the minimization problem from 8.2 with an appropriately-chosen diffusion tensor.

8.2.2. MULTIPATCH WITHOUT AUXILIARY VARIABLES

The non-affine nature of the pullbacks introduced in Section 8.2.1 prevent a computationally efficient approach based on the techniques from Chapter 3. Besides aspects of computational efficiency, an approach that avoids auxiliary variables may serve the prospect of combining multipatch with unstructured spline technologies, such as THB-splines. Preliminary numerical evidence suggests that the approach from Section 8.2.1 (and thus Chapter 3) is subject to the same inf-sup stability requirement as, for instance, the Stokes problem [BBF+13]. This justifies selecting the auxiliary variable spline space as uniformly h-refined with respect to the primal space on each macro element [BS13]. To the best of our knowledge, the only investigation of the inf-sub stability requirements corresponding to the isogeometric treatment of the Stokes problem with THB-splines is given in [BJ18]. The findings from [BBF+13] and [BJ18] suggests that, in an approach that combines multipatch and THB-splines, an h-refined auxiliary space is stable. However, due to the unstructured nature of THB-splines, the separability of the matrices from (3.17) and (3.18) is lost. This prevents eliminating the auxiliary variables from the problem formulation or necessitates direct inversion of the associated matrices using, for instance, a sparse factorization.

Here, an approach that employs a globally $C^0(\hat{\Omega})$-continuous spline basis, while coupling higher-order patch interface derivatives using discontinuous Galerkin (DG) [CKS12, LMMT15] techniques, may be a remedy. Not only does a DG-based approach avoid artificially inflating the problem size, it also requires no separate treatment of structured and unstructured spline spaces. A number of recent studies that investigate the finite element treatment of second-order elliptic equations in nonvariational form, which may be applicable to a linearization of the governing EGG equations, have appeared in the literature. To the best of our knowledge, Lakkis et al. made the first contribution in [LP11]. The approach introduces auxiliary variables for the Hessian instead of the gradient of the primal variable. C^0-DG schemes for classical FEA applications are developed in [DP13] and [FHN17], where interior penalty terms are employed for suppressing unphysical flux jumps across element edges. In an IGA-context, such penalizations may then be restricted to the patch interfaces, substantially reducing the number of DOFs that require special DG-based treatment compared to an approach that employs classical FEA elements. Another potential advantage is weakly imposing the Dirichlet boundary data, which simplifies the expression for the gradient of the objective function in Section 5.5. As such a penalty term operates on the zeroth derivative of the mapping, it can be disregarded for interior interfaces.

A further possibility is a $H^2(\hat{\Omega})$-conforming multipatch approach. This requires constructing a finite-dimensional spline basis $\mathcal{V}_h \subset H^2(\hat{\Omega})$ on the (possibly unstructured) multipatch domain $\hat{\Omega}$. To achieve this, in particular, we mention the techniques developed by Toshniwal et al. in [TSH17]. The approach combines THB-refinement by the patch interfaces and extraordinary vertices with the coupling of DOFs, resulting in a basis of higher regularity over $\hat{\Omega}$. The techniques from Chapter 4 are straightforwardly

extended to multipatch domains by discretizing the weak form over $\hat{\Omega}$ using the coupled THB-basis. An algorithmically convenient implementation prolongs the vector of weights from the coupled to the uncoupled basis, which is then used to assemble the residual in the usual way. Figure 8.3 shows the parameterization of a screw geometry

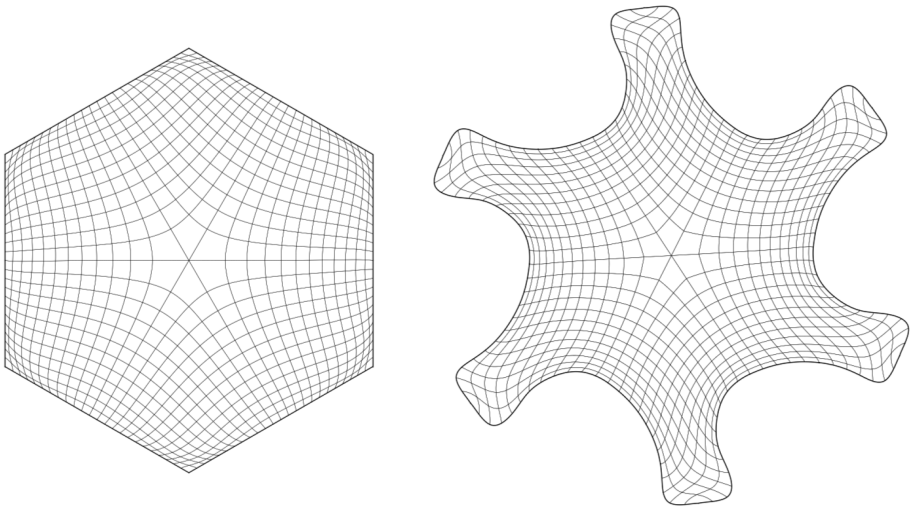


Figure 8.3: Screw parameterization that avoids auxiliary variables with the techniques from [TSH17].

along with the reparameterized domain, which is comprised of six patches. The approach avoids auxiliary variables by employing a basis $\mathcal{V}_h \subset H^2(\hat{\Omega})$ using the techniques from [TSH17]. It requires no DG-based coupling and is compatible with the computational approaches proposed in Section 4.2.

8.2.3. SPACE-TIME IGA FOR SWEPT SURFACES

As demonstrated in Section 2.10.2 and Chapters 6 and 7, the planar EGG equations enable parameterizing the interior of solids $\Omega \subset \mathbb{R}^3$ by sweeping a family of $\Omega^z \subset \mathbb{R}^2$ in z-direction. In the case of twin-screw compressors / extruders, the planar shells are a function of the rotational angle θ, i.e., $\Omega^z = \Omega^{z(\theta)}$, parameterized by the $\mathbf{x}^\theta : \hat{\Omega} \to \Omega^{z(\theta)}$, where $\hat{\Omega} \subset \mathbb{R}^2$ is static. The solid is then parameterized by

$$\mathbf{X}(\xi, \eta, \theta) = (\mathbf{x}^\theta(\xi, \eta), z(\theta))^T, \tag{8.3}$$

where $(\xi, \eta, \theta) \in \hat{\Omega} \times \Theta$.
In this section, we conceptualize the generalization of this principle to arbitrary swept surfaces and propose a *space-time* solution strategy.
Let $\mathbf{C} : (0,1) \to \mathbb{R}^3$ be a characteristic curve such that

$$\mathbf{n}(t) = \frac{\partial \mathbf{C}(t)}{\partial t} \quad \text{exists} \quad \forall t \in (0,1).$$

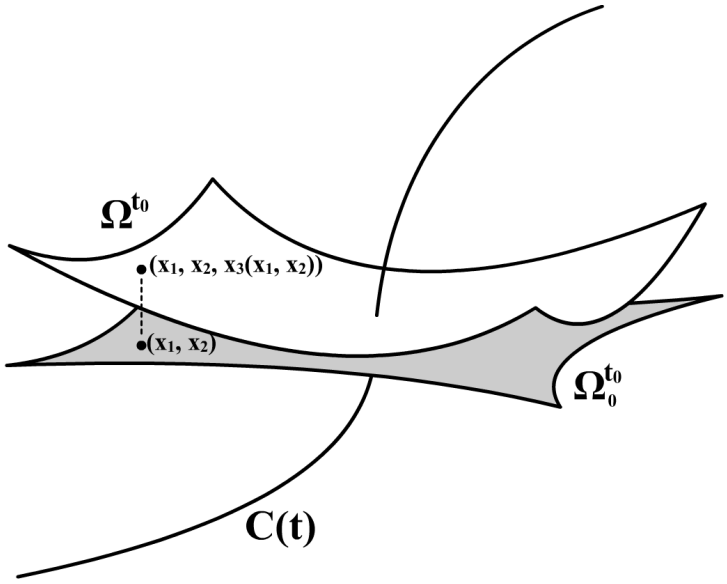


Figure 8.4: A shell $\Omega^{t_0} \subset \Omega$ whose coordinates are defined with respect to the normal plane of a characteristic curve $\mathbf{C}(t)$.

Furthermore, let $\Omega^t \subset \Omega$, $t \in (0,1)$ be a mutually disjoint family of two-dimensional shells. For given $t = t_0$, we assume that Ω^{t_0} possesses a preimage $\Omega_0^{t_0}$ with respect to a local chart $\gamma^{t_0} : \Omega_0^{t_0} \to \Omega^{t_0}$ of the form $(x_1, x_2) \to (x_1, x_2, x_3(x_1, x_2))$, where each $\Omega_0^{t_0}$ is topologically equivalent to $\hat{\Omega}$. Here, (x_1, x_2, x_3) is defined with respect to the coordinate system centered at $\mathbf{C}(t_0)$ and the plane with normal vector $\mathbf{n}(t_0)$, which locally defines the x_3-direction (see Figure 8.4). Ω^{t_0} constitutes a two-dimensional manifold in $\mathbb{R}^3$ and for fixed $t = t_0$, the planar EGG equations can be utilized to approximate a parameterization $\mathbf{X}|_{t=t_0} : \hat{\Omega} \to \Omega^{t_0}$ of the form

$$\mathbf{X}|_{t=t_0} = \left(\mathbf{x}^{t_0}(\xi,\eta), z\left(\mathbf{x}^{t_0}(\xi,\eta), t_0\right)\right)^T, \tag{8.4}$$

where $\mathbf{x}^{t_0} : \hat{\Omega} \to \Omega_0^{t_0}$ parameterizes $\Omega_0^{t_0}$ and $z(\cdot,\cdot,t_0)$ the x_3-coordinate with respect to the normal plane at $t = t_0$. The z-dependence results in an additional convective flux term upon pullback of the EGG equations over Ω^{t_0} into $\Omega_0^{t_0}$ [TSW98, Chapter 4]. We denote the operator that represents the corresponding PDE problem by $\mathscr{L}_{t,z}$ (note that hereby, z is regarded as having a t-dependence). This leads to a problem of the form

$$\mathscr{L}_{t,z}\left(\mathbf{X}(\xi,\eta,t)\right) = \mathbf{0} \quad \text{in} \quad \hat{\Omega} \times T, \quad \text{s.t.} \quad \mathbf{X}|_{\partial(\hat{\Omega}\times T)} = \partial\Omega, \tag{8.5}$$

where $T = (0,1)$. Given a finite-dimensional trivariate spline space $\mathcal{V}_h$ and assuming that

$$\mathbf{X}_h|_{\partial(\hat{\Omega}\times T)} = \partial\Omega \quad \text{with} \quad \mathbf{X}_h|_{\partial(\hat{\Omega}\times T)} \in \left(\mathcal{V}_h \setminus \mathcal{V}_h^\circ\right)^3,$$

a possible discretization of (8.5) reads:

$$\begin{aligned}
&\text{find } \mathbf{X}_h \in \mathcal{V}_h^3 \quad \text{s.t.} \quad \mathbf{X}_h|_{\partial(\hat{\Omega}\times T)} = \partial\Omega \quad \text{and} \\
&\int_{\hat{\Omega}\times T} \boldsymbol{\tau}(\boldsymbol{\sigma}_h) \cdot \mathscr{L}_{t,z}(\mathbf{X}_h)\,\mathrm{d}V + \int_{\hat{\Omega}\times T} \phi_h\,(Z_h - z(X_h, Y_h, t))\,\mathrm{d}V = 0, \\
&\forall (\boldsymbol{\sigma}_h, \phi_h) \in (\mathcal{V}_h^\circ)^2 \times \mathcal{V}_h,
\end{aligned} \tag{8.6}$$

where $\boldsymbol{\tau} : \left(\mathcal{V}_h^\circ\right)^2 \to L_2(\hat{\Omega}\times T, \mathbb{R}^2)$ and $\mathbf{X}_h = (X_h, Y_h, Z_h)^T$. To reduce memory requirements (which is particularly important in volumetric applications), we recommend solving (8.6) using a matrix-free Newton-Krylov algorithm, as in the preceding chapters.
In $\mathbb{R}^3$, accompanying Newton-Krylov with a preconditioner is recommendable. Here, we take inspiration from Chapters 6 and 7, where a volumetric problem is reduced to a sequence of planar problems by employing a spline space $\hat{\mathcal{V}}_h$ of polynomial order $p = 1$ in t-direction. Let $\mathcal{V}_h$ result from the knot vectors Ξ^p, $\mathscr{H}^p$ and $\mathscr{Z}^p$, with $p > 1$, while $\hat{\mathcal{V}}_h$ results from Ξ^p, $\mathscr{H}^p$ and $\hat{\mathscr{Z}}^1$. Chapters 6 and 7 discretize (8.5) using a collocation technique in t (and IGA in (ξ,η)). Putting the collocation points into the $\zeta_k \in \hat{\mathscr{Z}}^1$ (without knot repetitions) and discretizing in space as in the planar case, the problem breaks apart into a sequence of mutually independent planar problems. Hence, the Jacobian of the system is block diagonal

$$\hat{J} = \begin{pmatrix} \ddots & & \emptyset \\ & \hat{J}_k & \\ \emptyset & & \ddots \end{pmatrix} \tag{8.7}$$

with $|\{\ldots,\zeta_k,\ldots\}|$ blocks each corresponding to the Jacobian $\hat{J}_k$ of a planar problem. Hence, assembly and inversion of $\hat{J}$ is cheap compared to operating on a volumetric problem. We define the prolongation operator

$$T : \left[\hat{\mathcal{V}}_h\right] \to [\mathcal{V}_h], \quad \text{which is of the form} \quad T = I^{a\times a} \otimes I^{b\times b} \otimes T^{\mathscr{Z}},$$

for some $\{a, b\} \subset \mathbb{Z}$. A restriction operator

$$\hat{T} : [\mathcal{V}_h] \to \left[\hat{\mathcal{V}}_h\right], \quad \text{with} \quad \hat{T} = I^{a\times a} \otimes I^{b\times b} \otimes \hat{T}^{\mathscr{Z}}$$

results from taking $\hat{T}^{\mathscr{Z}}$ as the Moore-Penrose pseudoinverse of $T^{\mathscr{Z}}$. The collocation along with the tuple $(T, \hat{T})$ may then serve as a cost-effective way of preconditioning the space-time problem, thanks to the block diagonal nature of $\hat{J}$.
When $\hat{\Omega}$ is a multipatch domain, we introduce auxiliary variables as in Chapter 3. Note that the parametric properties of the solution of (8.6) can be tuned by introducing a control mapping

$$\mathbf{s} : \hat{\Omega}\times T \to \hat{\Omega}\times T,$$

which can for instance be used to achieve orthogonality at the boundaries (see Section 4.4). Here, reparameterization is conveniently accomplished by replacing derivatives with respect to (ξ,η,t) by derivatives with respect to the $(s_1, s_2, s_3)^T \equiv \mathbf{s}$.

REFERENCES

[BBF+13] Daniele Boffi, Franco Brezzi, Michel Fortin, et al. *Mixed finite element methods and applications*, volume 44. Springer, 2013.

[BJ18] Andrea Bressan and Bert Jüttler. Inf–sup stability of isogeometric taylor–hood and sub-grid methods for the stokes problem with hierarchical splines. *IMA Journal of Numerical Analysis*, 38(2):955–975, 2018.

[BS13] Andrea Bressan and Giancarlo Sangalli. Isogeometric discretizations of the stokes problem: stability analysis by the macroelement technique. *IMA Journal of Numerical Analysis*, 33(2):629–651, 2013.

[CKS12] Bernardo Cockburn, George E Karniadakis, and Chi-Wang Shu. *Discontinuous Galerkin methods: theory, computation and applications*, volume 11. Springer Science & Business Media, 2012.

[DP13] Andreas Dedner and Tristan Pryer. Discontinuous galerkin methods for non-variational problems. *arXiv preprint arXiv:1304.2265*, 2013.

[FHN17] Xiaobing Feng, Lauren Hennings, and Michael Neilan. Finite element methods for second order linear elliptic partial differential equations in non-divergence form. *Mathematics of Computation*, 86(307):2025–2051, 2017.

[LMMT15] Ulrich Langer, Angelos Mantzaflaris, Stephen E Moore, and Ioannis Toulopoulos. Multipatch discontinuous galerkin isogeometric analysis. In *Isogeometric Analysis and Applications 2014*, pages 1–32. Springer, 2015.

[LP11] Omar Lakkis and Tristan Pryer. A finite element method for second order nonvariational elliptic problems. *SIAM Journal on Scientific Computing*, 33(2):786–801, 2011.

[TSH17] Deepesh Toshniwal, Hendrik Speleers, and Thomas JR Hughes. Smooth cubic spline spaces on unstructured quadrilateral meshes with particular emphasis on extraordinary points: Geometric design and isogeometric analysis considerations. *Computer Methods in Applied Mechanics and Engineering*, 327:411–458, 2017.

[TSW98] Joe F Thompson, Bharat K Soni, and Nigel P Weatherill. *Handbook of grid generation*. CRC press, 1998.

LIST OF PUBLICATIONS

- **Hinz J.**, Jaeschke A., Möller M. & Vuik C. The Role of PDE-Based Parameterization Techniques in Gradient-Based IGA Shape Optimization Applications. Submitted to: *Computer Methods in Applied Mechanics and Engineering.*
- **Hinz J.**, Abdelmalik M., Möller M. Goal-Oriented Adaptive THB-Spline Schemes for PDE-Based Planar Parameterization.
- Shamanskiy A., Gfrerer M., **Hinz J.** & Simeon B. Isogeometric Parameterization Inspired by Large Elastic Deformation. In *Computer Methods in Applied Mechanics and Engineering.*
- **Hinz J.**, Helmig J., Möller M. & Elgeti, S. (2019). Boundary-Conforming Finite Element Methods for Twin-Screw Extruders using Spline-Based Parameterization Techniques. In *Computer Methods in Applied Mechanics and Engineering (2019): 112740.*
- **Hinz J.**, van Zwieten J., Möller M. & Vermolen F. (2019). Isogeometric Analysis of the Gray-Scott Reaction-Diffusion Equations for Pattern Formation on Evolving Surfaces and Applications to Human Gyrification.
- **Hinz J.**, Möller M. & Vuik C. (2019). An IGA Framework for PDE-Based Planar Parameterization on Convex Multipatch Domains.
- **Hinz J.**, Möller M. & Vuik C. (2018, September). Spline-Based Parameterization Techniques for Twin-Screw Machine Geometries. In *IOP Conference Series: Materials Science and Engineering* (Vol. 425, No. 1, p. 012030).
- Möller M. & **Hinz J.** (2018, September). Isogeometric Analysis Framework for the Numerical Simulation of Rotary Screw Machines. I. General Concept and Early Applications. In *IOP Conference Series: Materials Science and Engineering* (Vol. 425, No. 1, p. 012032). IOP Publishing.
- **Hinz J.**, Möller M. & Vuik C. Spline-Based Meshing Techniques for Industrial Applications.
- **Hinz J.**, Möller M. & Vuik C. (2018). Elliptic Grid Generation Techniques in the Framework of Isogeometric Analysis Applications. Computer Aided Geometric Design, 65, 48-75.