UNIVERSITY OF CALIFORNIA SAN DIEGO

Structural Analysis and Design Optimization of Non-matching Isogeometric Shells

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Engineering Sciences (Mechanical Engineering)

by

Han Zhao

Committee in charge:

Professor Jiun-Shyan Chen, Chair
Professor John T. Hwang
Professor H. Alicia Kim
Professor Shabnam J. Semnani

2024

The Dissertation of Han Zhao is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2024

DEDICATION

To my family

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First, I would like to express my sincere gratitude to my advisor, Professor J. S. Chen, for his constant guidance, support, and encouragement throughout my Ph.D. journey. His expertise, patience, and insightful feedback have been invaluable. I am deeply grateful for the opportunity to be part of Professor Chen's group and learn from his extensive knowledge and dedication to research. I am also immensely thankful to Dr. David Kamensky for his excellent and patient mentoring, and for introducing me to the world of isogeometric analysis.

I would like to acknowledge my defense and candidacy committee members, Professor John T. Hwang, Professor H. Alicia Kim, and Professor Shabnam J. Semnani, for their helpful feedback and serving on my committee. I also want to thank Professor John T. Hwang for his research suggestions and excellent collaboration. I would like to thank Professor Nicholas Boechler for his guidance and support during my initial time at UCSD. A special thanks to Professor Ming-Chen Hsu for his support and great discussions.

I would like to extend my appreciation to the members and friends of our research group, Ru Xiang, Jennifer Fromm, Kristen Susuki, Ryan Schlinkman, Yanran Wang, Samuel Casebolt, Dr. Karan Taneja, Dr. Jonghyuk Baek, Dr. Marco Pasetto, Joshua Krokowski, Sebastiaan van Schie, for their collaboration, company, and for making my Ph.D. experience both productive and enjoyable.

I am also thankful to my colleagues and friends at UCSD, Dr. Maroun Abi Ghanem, Dr. Reza Behrou, Michael Warner, Marius Ruh, Anugrah Jo Joshy, Andrew Fletcher, Xiangbei Liu, Dr. Darshan Sarojini, Tianshi Feng, Ning Li, Dr. Zichen Zhang, Dr. Li Tan, Dr. Erbin Qiu, Dr. Liang Ji, Yida Wu, Aobo Yang, Jie Feng, Robert Chambers, Putian He and many others, whose support and friendship have greatly motivated me throughout this journey.

Finally, I would like to express my sincere gratitude to my mom, dad, sister, and grandparents for their unconditional love and always believing in me. I would not have been able to accomplish this without them. I would like to thank Yunzhao Qiao for her companionship and constant encouragement.

The support from the National Aeronautics and Space Administration (NASA) is greatly appreciated.

Portions of Chapters 1, 2, 3, 6, 7, and 8 have been published in "H. Zhao, X. Liu, A. H. Fletcher, R. Xiang, J. T. Hwang, and D. Kamensky. An open-source framework for coupling non-matching isogeometric shells with application to aerospace structures *Computers & Mathematics with Applications*, 111:109–123, 2022." The dissertation author is the primary investigator and author of this paper.

Portions of Chapters 1, 2, 4, 6, and 8 have been published in "H. Zhao, D. Kamensky, J. T. Hwang, and J. S. Chen. Automated shape and thickness optimization for non-matching isogeometric shells using free-form deformation. *Engineering with Computers*, 1-24, 2024." The dissertation author is the primary investigator and author of this paper.

Portions of Chapters 1, 2, 3, 5, 6, and 8 have been published in "H. Zhao, J. T. Hwang, and J. S. Chen. Shape optimization of non-matching isogeometric shells with moving intersections. *Computer Methods in Applied Mechanics and Engineering*, 431:117322, 2024." The dissertation author is the primary investigator and author of this paper.

Portions of Chapters 1, 2, 7, and 8 are currently being prepared for submission for publication of the material. "H. Zhao, J. T. Hwang, and J. S. Chen. Open-source shape optimization for isogeometric shells using FEniCS and OpenMDAO". The dissertation author was the primary investigator of this material.

VITA

2024	Ph.D. in Engineering Sciences (Mechanical Engineering), University of California San Diego
2020	M.S. in Engineering Sciences (Mechanical Engineering), University of California San Diego
2018	B.S. in Vehicle Engineering, Liaoning Technical University

PUBLICATIONS

JOURNAL ARTICLES

- H. Zhao, J. T. Hwang, and J. S. Chen. Open-source shape optimization for isogeometric shells using FEniCS and OpenMDAO. In preparation.
- H. Zhao, J. T. Hwang, and J. S. Chen. Shape optimization of non-matching isogeometric shells with moving intersections. *Computer Methods in Applied Mechanics and Engineering*, 431:117322, 2024.
- H. Zhao, D. Kamensky, J. T. Hwang, and J. S. Chen. Automated shape and thickness optimization for non-matching isogeometric shells using free-form deformation. *Engineering with Computers*, 1-24, 2024.
- J. E. Fromm, N. Wunsch, R. Xiang, H. Zhao, K. Maute, J. A. Evans, and D. Kamensky. Interpolation-based immersed finite element and isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 405:115890, 2023.
- G. E. Neighbor, H. Zhao, M. Saraeian, M.-C. Hsu, and D. Kamensky. Leveraging code generation for transparent immersogeometric fluid–structure interaction analysis on deforming domains. *Engineering with Computers*, 39(2):1019–1040, 2023.
- H. Zhao, X. Liu, A. H. Fletcher, R. Xiang, J. T. Hwang, and D. Kamensky. An open-source framework for coupling non-matching isogeometric shells with application to aerospace structures. *Computers & Mathematics with Applications*, 111:109–123, 2022.
- X. Liu, C. Du, X. Fu, H. Zhao, J. Zhang, and X. Yang. Wear analysis and performance optimization of drum blade in mining coal gangue with shearer. *Engineering Failure Analysis*, 128:105542, 2021.
- M. A. Ghanem, A. Khanolkar, H. Zhao, and N. Boechler. Nanocontact tailoring via microlensing enables giant postfabrication mesoscopic tuning in a self-assembled ultrasonic metamaterial. *Advanced Functional Materials*, 30(10):1909217, 2020.

CONFERENCE PROCEEDINGS

- D. Sarojini, M. L. Ruh, J. Yan, L. Scotzniovsky, N. C. Orndorff, R. Xiang, H. Zhao, J. Krokowski, M. Warner, S. P. van Schie, A. Cronk, A. T. Guibert, J. T. Chambers, L. Wolfe, R. Doring, R. Despins, C. Joseph, R. Anderson, A. Ning, H. Gill, S. Lee, Z. Cheng, Z. Cao, C. Mi, Y. S. Meng, C. Silva, J. S. Chen, A. A. Kim, J. T. Hwang. Review of Computational Models for Large-Scale MDAO of Urban Air Mobility Concepts. *AIAA SciTech 2024 Forum*, 2024.

- M. L. Ruh, A. Fletcher, D.Sarojini, M. Sperry, J. Yan, L. Scotzniovsky, S. P. van Schie, M. Warner, N. C. Orndorff, R. Xiang, A. J. Joshy, H. Zhao, J. Krokowski, H. Gill, S. Lee, Z. Cheng, Z. Cao, C. Mi, C. Silva, L. Wolfe, J. S. Chen, J. T. Hwang. Large-scale multidisciplinary design optimization of a NASA air taxi concept using a comprehensive physics-based system model. *AIAA SciTech 2024 Forum*, 2024.

- S. P. van Schie, H. Zhao, J. Yan, R. Xiang, J. T. Hwang, and D. Kamensky. Solver-independent aeroelastic coupling for large-scale multidisciplinary design optimization. *AIAA SciTech 2023 Forum*, 2023.

ABSTRACT OF THE DISSERTATION

Structural Analysis and Design Optimization of Non-matching Isogeometric Shells

by

Han Zhao

Doctor of Philosophy in Engineering Sciences (Mechanical Engineering)

University of California San Diego, 2024

Professor Jiun-Shyan Chen, Chair

Isogeometric analysis (IGA) has emerged as a powerful approach in the field of structural analysis, benefiting from the seamless integration between the computer-aided design (CAD) geometry and the analysis model by employing non-uniform rational B-splines (NURBS) or other types of splines as basis functions. The spline basis functions naturally satisfy the C^1 continuity making it particularly suitable for the approximation of the Kirchhoff–Love shell formulation for thin shells. Despite its advantages, the application of IGA to structural analysis and design optimization of complex shell CAD geometries consisting of multiple non-matching spline patches remains challenging due to arbitrary surface intersections. To achieve the streamlined design-analysis-optimization workflow, effective coupling of intersecting shell patches in

structural analysis and special handling to maintain intersections during design optimization is essential.

In this work, a shell coupling algorithm is developed to directly analyze complex shell structures represented as collections of untrimmed NURBS patches. Shell patches are modeled mechanically as Kirchhoff–Love shells and discretized isogeometrically. Coupling of non-matching patches uses a penalty-based formulation, employing a series of topologically 1D, geometrically 2D quadrature meshes. The quadrature meshes act as integration domains for the penalty energy to preserve displacement and rotational continuities at patch intersections. For design optimization, a free-form deformation (FFD)-based formulation is proposed for shape and thickness optimization of non-matching shell structures, ensuring compatibility of design variables at patch intersections throughout the optimization process. Lagrange extraction is employed to link control points associated with the B-spline FFD block and shell patches, and the extraction operators are also used to represent NURBS functions by Lagrangian bases in IGA analysis. Analytical sensitivities are derived to facilitate efficient gradient-based optimization algorithms. Additionally, a novel method for shape optimization of non-matching isogeometric shells incorporating intersection movement is introduced, allowing shell patches to move independently during shape updates. This flexibility is achieved by an implicit state function, with analytical sensitivities derived for the relative movement of shell patches. The differentiable intersections expand the design space and overcome challenges associated with large mesh distortion when optimal shapes involve significant movement of patch intersections in physical space. Shell patches are represented by the NURBS bases during the optimization process, enabling efficient integration of analysis and design models, and allowing for the multilevel design concept.

The proposed structural analysis algorithm and design optimization methods are validated through various benchmark problems and applied to practical aircraft wings, demonstrating their effectiveness for complex shell structures.

Chapter 1

Introduction

1.1 Motivation

1.1.1 Integration of design and analysis for complex shell structures

It is well-established that the analysis bottleneck in exploring geometrical design spaces is mesh generation [76]. The field of IGA [88, 45] aims to directly analyze spline-based geometry representations used in computer-aided design programs, bypassing mesh generation entirely. A comparison of discretization schemes for an electric vertical take-off and landing (eVTOL) aircraft wing between classical finite element method (FEM) and IGA is shown in Figure 1.1. The spline function spaces used in CAD also have mathematical properties that make them particularly useful for approximating solutions to partial differential equations (PDEs), independent of IGA's initial goal of streamlining design-through-analysis. In particular, smooth polynomial splines have superior approximation power per degree-of-freedom relative to traditional high-order finite elements [49], and the additional regularity permits Bubnov–Galerkin approximations of high-order PDEs.

Shell structures exhibit exceptional stiffness and high strength-to-self-weight ratios, making them widely used in various engineering fields, including aerospace, automotive, and marine engineering [51]. These structures are characterized by thin and curved geometries, which require precise modeling and analysis to accurately predict their structural behavior. The Kirchhoff–Love shell theory [105, 103], which assumes negligible transverse shear strains,

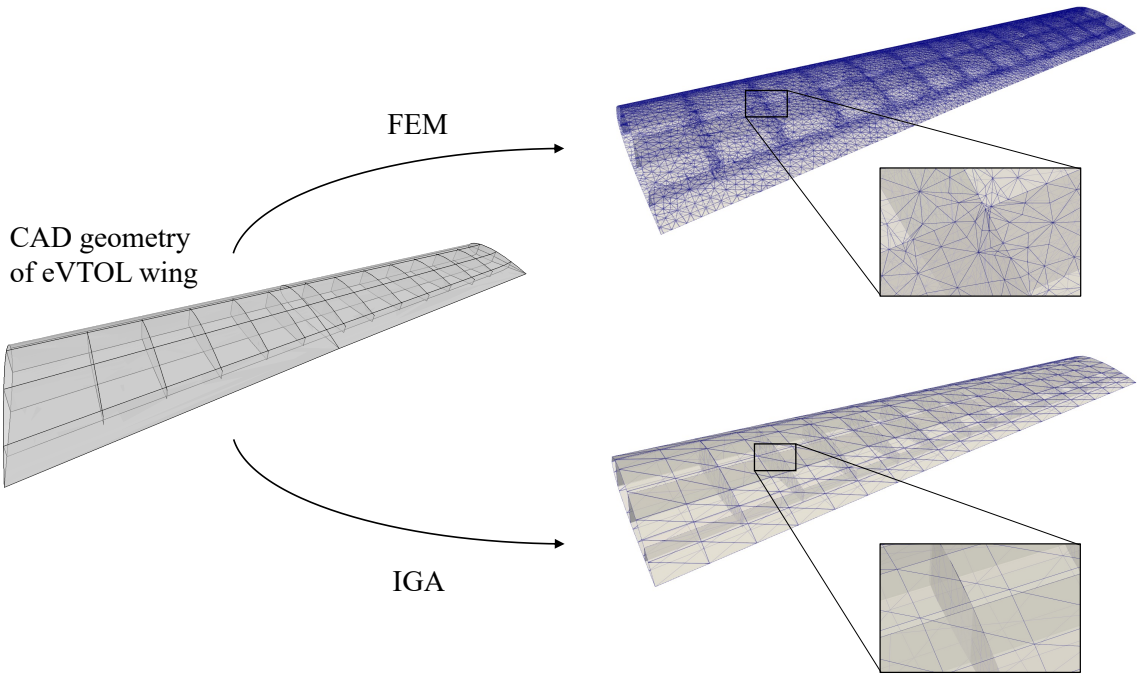


Figure 1.1. Discretization schemes comparison of an eVTOL wing using FEM and IGA. High-quality connected meshes generation is required in the classical FEM, while mesh generation is circumvented due to the use of splines as basis functions in IGA.

is particularly suited for thin shell structures but requires C^1 continuity in the displacement field. Achieving higher-order continuity is intractable in classical FEM, as it requires specially designed elements. In contrast, the spline basis functions used in IGA naturally provide the C^1 continuity across element boundaries, making IGA an ideal approach for approximating solutions to Kirchhoff–Love shell problems.

However, challenges arise when applying IGA to practical CAD geometries composed of multiple non-matching spline patches. Complex shell models often feature arbitrary surface intersections, as both displacement and rotational continuities need to be maintained at these intersections. Figure 1.2 shows a pair of intersecting shell patches with a single intersection. To address these challenges, coupling methods must be developed to ensure displacement and angular compatibility at surface intersections, which are essential for accurate displacement predictions. Furthermore, with the implementation of coupling methods, complex shell CAD geometries are directly available for structural analysis without the need for FE mesh generation,

thereby fulfilling the potential of IGA to streamline design through analysis.

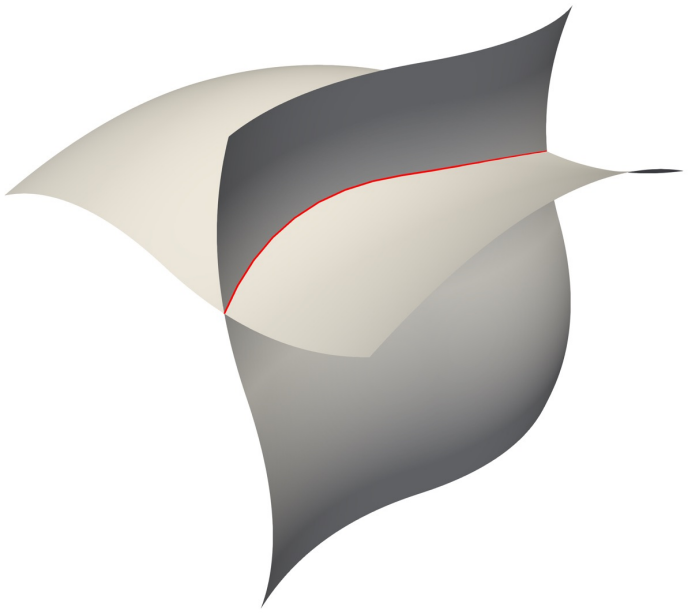


Figure 1.2. An illustrative example of a pair of shell patches with a single surface–surface intersection, where displacement and angular compatibility need to be maintained in structural analysis.

1.1.2 Design optimization of complex shell structures

The performance of shell structures is significantly influenced by their geometric and material properties. Thus, structural optimization plays a critical role in enhancing structural performance. Structural optimization for shell structures can be categorized into four main types: material optimization, sizing optimization, shape optimization, and topology optimization [103]. Topology optimization [48, 17] focuses on the optimal distribution of material within the design domain. It has gained significant attention in the design of shell structures, leading to many lightweight and highly efficient designs. However, the non-intuitive and complex material layouts resulting from topology optimization pose additional complexities for manufacturing, necessitating additional constraints and considerations for feasible production. Material optimization [155] explores the optimal selection of materials with specific properties that best meet the design requirements, which is particularly beneficial for composite structures. In shell design,

this approach allows for the use of different materials or layouts in each layer to optimize the overall structural performance. Sizing optimization [102] for shell structures targets the optimal distribution of thickness to achieve lightweight and efficient designs. This approach directly addresses the challenge of reducing material usage while maintaining structural performance. Moreover, shape optimization [21] aims to determine the optimal geometry of shell structures, where applied loads are primarily supported by membrane forces and the bending moments are minimized. Unlike material and thickness optimizations, shape optimization alters the shell patches' geometry by adjusting their curvature, orientation, and layouts. This approach can lead to significant improvement of shell structures' mechanical characteristics, enhancing their overall performance and efficiency.

Structural analysis provides essential insights into the structural performance of updated configurations during design optimization. Therefore, the accuracy and efficiency of structural analysis are critical for achieving optimal designs. In the conventional approach, FEM is commonly used for structural analysis throughout the optimization process. However, inherent difficulties arise in the FEM-based design optimization for shell structures. In the case of the Kirchhoff–Love shells, the requirement of C^1 continuity is difficult to satisfy using FEM, necessitating specialized elements. Additionally, FEM is sensitive to mesh quality, and variations in element size, shape, and aspect ratio can result in ill-conditioned numerical systems, adversely affecting analysis accuracy and optimization results. Shape changes in shell structures often demand re-meshing to maintain mesh quality, which is intractable and time-consuming in practice. Moreover, each iteration of generating the FE mesh from the updated shell design introduces geometric errors, compromising the accurate representation of the actual geometry and reducing the accuracy of the FEM-based optimization solution.

On the contrary, IGA offers a robust and accurate approach to the design optimization of shell structures by integrating CAD models with analysis models directly, overcoming many limitations faced in classical FEM. In IGA-based shape optimization, the coordinates of control points that define the shell CAD geometry serve as design variables. The updated design is

achieved by modifying these control points, allowing for the direct evaluation of structural response without FE mesh generation, thus eliminating geometric errors. Higher analysis accuracy and effective shape sensitivities can also be achieved using spline basis functions, which ensure the quality of optimal solutions and efficient convergence. Furthermore, the geometry preservation properties in *p*-, *h*-, and *k*-refinements enable multilevel design, where the dimension of design space can be selected independent of the highly refined analysis model, enhancing convergence and realistic designs without compromising the geometric accuracy.

Despite the exceptional properties of IGA in the design optimization of shell structures, the handling of patch intersections, where shell patches do not share conforming degrees of freedom (DoFs), in complex shell geometries remains an active research direction. Ensuring robust and accurate structural analysis in the optimization loop is crucial, as compatibility conditions need to be maintained at patch intersections. Without proper handling, shell patches may become disjointed during shape updates, leading to unrealistic designs. Additionally, large movement of patch intersections when optimizing shell layouts can result in extremely skinny shell elements, particularly if intersecting shell patches are optimized concurrently. Advanced numerical methods are necessary to address these challenges in shape optimization of shell structures using IGA so that the entire design-analysis-optimization workflow can be streamlined and automated, thereby, facilitating the conceptual development of novel shell structures in engineering design. Establishing the workflow will enable engineers to fully leverage the potential of IGA.

1.2 Objectives

The objective of this work is to develop numerical methods and establish a streamlined workflow for the design-analysis-optimization process of complex shell structures, with applications to aircraft wing designs. The main contributions of this dissertation are summarized as follows:

1. Development of a penalty-based coupling algorithm based on the formulation proposed in [77] for isogeometric shell structures consisting of collections of untrimmed NURBS surfaces. Shell patches are modeled using Kirchhoff–Love shell theory, where the C^1 continuity requirement is automatically satisfied by NURBS basis functions. With the coupling algorithm, displacement and angular compatibility are maintained by integrating the penalty energy along patch intersections, which involves the displacements and curvilinear basis vectors of the intersecting shell patches. The penalty energy integration domains are represented by geometrically 2D, topologically 1D quadrature meshes in the parametric space. With automated shell coupling and isogeometrically discretized shell patches, CAD geometries of complex shell structures can be used for analysis directly without the need for FE mesh generation, thus streamlining the design-through-analysis workflow.

2. Development of an FFD-based [152] shape and thickness optimization approach for non-matching isogeometric shells. The shape of the shell structure is updated through a trivariate B-spline FFD block that encompasses the entire shell structure. The FFD block modifies the Lagrange nodal points of all shell patches concurrently, preserving the surface–surface intersections. Subsequently, the resulting NURBS surfaces of shells are obtained using the Lagrange extraction technique [148], which is implemented in IGA using FE subroutines. Meanwhile, this approach is applicable to shell thickness optimization where the thickness distribution is continuous at patch intersections. By integrating these two design variables, simultaneous shape and thickness optimization for non-matching shells can be effectively achieved. This combined optimization approach enables the exploration of complex design spaces while preserving the compatibility of the non-matching shell structure.

3. Development of a general shape optimization approach for non-matching shell structures where intersecting shell patches are allowed to move independently and have relative movements. Parametric locations of intersections are no longer fixed in the optimization

process. The optimization process considers shape changes of selected sub-structures, allowing intersections to relocate to accommodate these changes. Control points of shell patches are optimized directly without additional operations, while the locations of the surface intersections are updated accordingly through an implicit relation between surface control points and intersections' parametric coordinates. This enables relative movement between shell patches without distorting the shell elements. The penalty method is employed to couple the non-matching shell patches in structural analysis. Throughout the optimization process, parametric coordinates of intersections are solved correspondingly when updating the geometry of shell patches. Sensitivities of the implicit relation and penalty residual with respect to intersections' parametric coordinates are derived to obtain the total derivative of the optimization problem and facilitate gradient-based optimization algorithms.

4. Implementation of the proposed methods mentioned above into open-source Python libraries to promote code transparency and contribute to the IGA-based design optimization community. The open-source implementations make use of the code generation capabilities and automatic differentiation provided by the FEniCS project [122] to streamline the design-analysis-optimization workflows for non-matching isogeometric shells. Assembly subroutines in FEniCS are used to automate the computation. Additionally, the optimization framework inherits the modular design from OpenMDAO [63], allowing for flexible multidisciplinary optimization. The open-source frameworks are verified through numerical benchmark problems and applied to real-world CAD geometries of aircraft wings, demonstrating their effectiveness for complex shell structures.

1.3 Outline

The remainder of the dissertation is outlined as follows. An overview of structural analysis and design optimization methods for complex shell structures, along with coupling methods

for non-matching shell patch intersections, are discussed in Chapter 2. The coupling algorithm using the penalty-based formulation for isogeometric Kirchhoff–Love shells is illustrated in Chapter 3. Chapter 4 presents the detailed formulations of the FFD-based thickness and shape optimization approach, integrated with the Lagrange extraction technique, for non-matching shell structures. In Chapter 5, a general shape optimization method involving moving intersections is demonstrated. Various benchmark problems with reference solutions are used to verify the proposed analysis and optimization frameworks. Chapter 6 showcases the application of the proposed methods to structural analysis and design optimization of practical aircraft wings, yielding promising analysis results and innovation designs for aerospace structures. The open-source implementation and usage of the analysis and optimization frameworks, based on FEniCS and OpenMDAO, are illustrated in Chapter 7. Finally, Chapter 8 provides concluding remarks on the proposed methods and discusses future research directions.

Acknowledgements

A portion of this chapter has been published in "H. Zhao, X. Liu, A. H. Fletcher, R. Xiang, J. T. Hwang, and D. Kamensky. An open-source framework for coupling non-matching isogeometric shells with application to aerospace structures *Computers & Mathematics with Applications*, 111:109–123, 2022." The dissertation author is the primary investigator and author of this paper.

A portion of this chapter has been published in "H. Zhao, D. Kamensky, J. T. Hwang, and J. S. Chen. Automated shape and thickness optimization for non-matching isogeometric shells using free-form deformation. *Engineering with Computers*, 1-24, 2024." The dissertation author is the primary investigator and author of this paper.

A portion of this chapter has been published in "H. Zhao, J. T. Hwang, and J. S. Chen. Shape optimization of non-matching isogeometric shells with moving intersections. *Computer Methods in Applied Mechanics and Engineering*, 431:117322, 2024." The dissertation author is

the primary investigator and author of this paper.

A portion of this chapter is currently being prepared for submission for publication in "H. Zhao, J. T. Hwang, and J. S. Chen. Open-source shape optimization for isogeometric shells using FEniCS and OpenMDAO". The dissertation author was the primary investigator of this material.

Chapter 2

Literature Review

This chapter reviews the basics of IGA and structural analysis of shell structures, discusses coupling methods for non-conforming intersections, and presents design optimization strategies of shell geometries.

2.1 Isogeometric analysis

IGA [88, 45] has gained increasing attention since its introduction in 2005, as it integrates the CAD and analysis models into a unified framework using spline basis functions, e.g., B-splines, NURBS [138], T-splines [153, 9], THB-splines [58], and U-splines [159]. IGA addresses the key bottleneck in classical FEA, which involves intricate and time-consuming FE mesh generation process [76]. Geometric errors are introduced in this process, particularly for complex geometries, and the analysis accuracy is highly sensitive to the quality of elements in FE meshes.

IGA was developed to eliminate the FE mesh generation by performing analysis directly on the input CAD geometries, thereby preserving the exact geometry in analysis and streamlining the numerical analysis workflow. Similar to the h-refinement and p-refinemenet in FEM, IGA has corresponding methods that are achieved by knot insertion and order elevation of NURBS, both of which maintain the original geometry without introducing geometric errors. Additionally, IGA uniquely offers k-refinement, achieved by order elevation followed by inserting unique knots, providing increased continuity of the basis functions, which is difficult to accomplish

in classical FEM. The smoothness of spline basis functions in IGA allows direct discretization of higher-order PDEs and has been successfully applied to models such as Kirchhoff–Love shells [103, 105, 106] and the Cahn–Hilliard phase field model [71]. Moreover, the spline basis functions exhibit excellent approximation properties per DoFs [49], making them highly effective for complex engineering applications.

2.2 Structural analysis of shell structures

Shell structures are a fundamental class of structural elements widely used in engineering applications due to their efficient load-carrying capabilities. Shell theory models the structural behavior of these structures under various load and boundary conditions, ensuring that shell designs meet the performance and safety requirements. A key assumption in shell theory is that the thickness is much smaller than other dimensions, allowing the geometry to be simplified and represented by its mid-surface. Two primary shell theories are commonly employed in structural analysis. The first one is the Reissner–Mindlin shell theory [14], which accounts for transverse shear strains and describes the deformed state using displacement and rotational fields. The Reissner–Mindlin shell theory only requires C^0 continuity for the solution, making it suitable for implementation with FEM. Due to its consideration of the transverse shear strains, the Reissner–Mindlin shell theory can be applied to thin to moderately thick shells. On the other hand, the Kirchhoff–Love shell theory [14] is ideal for very thin shells, where transverse shear strains are negligible. It assumes that cross-sections remain straight and normal to the mid-surface after deformation and thickness keeps constant, allowing for the deformed state to be described solely by the displacement field of the mid-surface. The Kirchhoff–Love shell theory leads to a fourth-order PDE and therefore requires C^1 continuity for the displacement. These theories provide the fundamental framework for analyzing shell structures.

Computational analysis of shell structures has been investigated extensively in mechanical and civil engineering. Various numerical methods have been developed and successfully applied

to industrial problems. FEM [87, 14] remains the industrial standard and is widely used in both academia and industry. In FEM, shell structures are divided into small, simple elements, such as triangles or quadrilaterals, with interconnected nodes that approximate the initial shell geometry. Then the weak form or variational equation of the shell's governing PDE is discretized and formulated into a system of linear equations using Galerkin approximation to solve for the unknowns. Due to the use of Lagrange polynomials as basis functions, standard FEM elements only provides C^0 continuity on element boundaries, making them suitable for Reissner–Mindlin shell models. However, careful selection of function spaces is needed to mitigate shear locking in very thin shells, a common numerical issue that can compromise analysis accuracy. Implementing FEM for Kirchhoff–Love shells poses additional complexities due to the C^1 continuity requirement, which standard FEM elements do not naturally support. Moreover, the generation of FE meshes remains a bottleneck in FEM applications, accounting for up to 80% of the total analysis time [76].

Another popular class of computational methods for shell structures is meshfree methods [32, 13], such as reproducing the kernel particle method (RKPM) [121, 120, 37, 40, 35, 36] and the element-free Galerkin (EFG) method [15, 124, 114, 115, 96]. Interconnected FE meshes are no longer required in these methods. Instead, they rely on a collection of discrete points with constructed shape functions, significantly simplifying the analysis workflow. Furthermore, the C^1 continuity in the Kirchhoff–Love shell theory can be easily satisfied in meshfree methods, making them suitable for thin shell analysis. Despite these advantages, meshfree methods tend to have higher computational costs compared to classical FEM. To address this, more advanced integration techniques [38, 39, 167, 141, 33, 80] can be considered to improve computational efficiency.

On the other hand, the requirement of the C^1 continuity across element boundaries in the Kirchhoff–Love shell theory is naturally satisfied without additional efforts in IGA. This makes IGA an attractive analysis method for Kirchhoff–Love shells and other higher-order PDEs without the additional complexities required in classical FEM. The seamless integration between

CAD models and analysis models simplifies the analysis workflow significantly since the CAD geometry can be analyzed directly. This integration also makes IGA a more ideal approach for shape optimization shell structures compared to other numerical methods. Isogeometric Kirchhoff–Love shells have been studied extensively in [105, 103, 106, 135, 149, 28, 30, 126], and have been applied across a variety of domains with excellent results, including analysis of wind turbine blades [10, 94, 12, 78], heart valve leaflets [100, 127, 99, 171, 95, 170, 97, 130], graphene sheets [154, 56, 57], and aerospace structures [81, 181, 74, 144]. Furthermore, various approaches [156, 64, 19, 27, 29, 65, 134, 145] have been proposed to address the membrane locking issues in Kirchhoff–Love shell problems.

2.3 Coupling methods for isogeometric shells

Multiple patches are typically required to model complex, realistic shell structures using NURBS surfaces. To make the CAD geometries with multiple patches directly available for structural analysis, coupling between adjacent NURBS patches to maintain displacement and angular compatibility across patch intersections becomes essential. The bending strip method [104] and the kinematic constraints [47] have been proposed for coupling NURBS surfaces with conforming discretizations. For isogeometric Kirchhoff–Love shells with non-matching intersections, a series of coupling techniques have been explored, including mortar methods [25, 85, 82], Nitsche-type methods [69, 68, 70, 18, 168, 176, 136, 31], projected super-penalty methods [44, 43], penalty methods [77, 116, 140, 181, 67], and embedded surfaces methods [83, 74]. Streamlined design through analysis workflow is achieved by employing multipath coupling in isogeometric shell structures, where displacement differences and rotational changes are eliminated at patch intersections. This approach completely bypasses the FE mesh generation within the workflow, thereby significantly reducing manual effort in analysis.

2.4 Design optimization for shell structures

A well-designed shell structure features excellent performance by distributing load through membrane forces while minimizing bending moments [20], with the mechanical characteristics significantly affected by its shape. Consequently, shape optimization plays a critical role in the development of novel shell structures. The unified model between geometric design and structural analysis in IGA renders particular advantages for shell shape optimization [166, 41, 72, 142, 117, 4], with many superior designs such as composite shells [129, 75], wind turbine blades [79, 78], and stiffened thin-wall structures [81, 74].

Traditionally, shape optimization relies on the FEM with parametric models [46]. However, the classical FEM-based approach encounters difficulties in the precise representation of the updated geometry and accurate solution of structural behavior [90]. Shape optimization using meshfree methods, such as RKPM-based design optimization [111, 108, 110, 109, 66, 34], offers an effective solution to many difficulties encountered in FEM-based shape optimization, particularly issues related to mesh quality and low-order continuous basis functions. RKPM bypasses these challenges by eliminating the dependency on FE meshes. In addition, RKPM demonstrates distinct advantages in topology optimization [132, 133, 131] with an exact representation of geometry and straightforward control of the order of continuity, ensuring smoother topology evolution during the optimization process. Despite these benefits, the meshfree method still requires additional steps to generate the point clouds that accurately represent the geometry and generation of RK shape functions.

Shape optimization using IGA addresses these challenges by directly performing structural analysis on the CAD model, circumventing the intermediate steps as in the FEM-based approach or the RKPM-based method. The geometric error between design and analysis models is eliminated and the continuity of the geometry is preserved by adjusting the coordinates of the control points during the optimization process. Nonetheless, updating complex geometries with multiple NURBS patches necessitates additional efforts to represent surface intersections

accurately. The works in [83, 24, 74] utilized the spline composition. This method involves combining a volume spline mapping, which is created by extruding a "master" surface to form a 3D block, and a surface spline mapping to align the stiffeners (sub-structures) with the "master" surface perfectly. The "master" surfaces need to be identified to perform the extrusion, and spline composition between volume and surface mappings is required in the former method. [180] employed the FFD technique [152] in conjunction with Lagrange extraction [148] to perform shape optimization for the non-matching shell patches while maintaining the intersection geometries. However, this method may lead to substantial distortion of elements when surface intersections undergo large movement. The recent work in [179] allows relative movement between intersection shell patches during shape optimization, circumventing the element distortion issue with moving intersections.

2.5 Open-source shape optimization of isogeometric shells

Code transparency in the field of design optimization has been gaining more interest. OpenMDAO [63], an open-source framework for multidisciplinary design optimization (MDO), uses a modular architecture that allows users to create custom models for different disciplines and integrate them with various optimization algorithms. Successful applications of OpenMDAO span aerospace engineering [92, 91, 2], wing energy [78], robotics [173, 118], and topology optimization [42, 174, 93]. The Python library CSDL [55] addresses large-scale MDO problems using a graph representation to automatically generate adjoint sensitivities. The CSDL-based Python library FEMO [169], coupled with the FEniCS project [123, 122, 3], was developed to solve partial differential equation (PDE)-constrained optimization problems, significantly reducing the coding effort for PDE components such as structural mechanics, fluid mechanics, and heat transfer, etc. Despite these advancements, an open-source design optimization framework using IGA has been lacking. This contribution fills that gap by developing an open-source Python library for shape optimization of complex shell structures using IGA, enabling researchers

to explore the benefits of IGA in shape optimization problems and advance structural design optimization.

In the proposed code framework [178], OpenMDAO is used as the optimization toolkit. For the IGA solver for structural analysis of non-matching shell structures, the open-source package PENGoLINS [181] is employed. PENGoLINS, a Python framework based on tIGAr [98], uses extraction techniques [23, 150, 148, 54, 53] to construct spline basis functions from Lagrange basis functions in the FEM solver FEniCS. tIGAr has been successfully applied in various fields [11, 158, 177, 175]. Additionally, an open-source fluid–structure interaction framework [97, 130] is developed based on tIGAr and shows a good application for prosthetic heart valve simulation with isogeometric leaflets. PENGoLINS employs a penalty-based formulation [181] to couple the non-matching isogeometric Kirchhoff–Love shells, which is automated with the code generation technology in FEniCS. The current code framework incorporates IGA for complex Kirchhoff–Love shells with a penalty formulation [77], and employs the FFD-based and moving intersection approaches [180, 179] to handle patch intersections. The modular architecture of OpenMDAO ensures that the open-source implementation of isogeometric shell shape optimization can readily couple with other disciplines and extend to more practical problems.

Acknowledgements

A portion of this chapter has been published in "H. Zhao, X. Liu, A. H. Fletcher, R. Xiang, J. T. Hwang, and D. Kamensky. An open-source framework for coupling non-matching isogeometric shells with application to aerospace structures *Computers & Mathematics with Applications*, 111:109–123, 2022." The dissertation author is the primary investigator and author of this paper.

A portion of this chapter has been published in "H. Zhao, D. Kamensky, J. T. Hwang, and J. S. Chen. Automated shape and thickness optimization for non-matching isogeometric shells using free-form deformation. *Engineering with Computers*, 1-24, 2024." The dissertation

author is the primary investigator and author of this paper.

A portion of this chapter has been published in "H. Zhao, J. T. Hwang, and J. S. Chen. Shape optimization of non-matching isogeometric shells with moving intersections. *Computer Methods in Applied Mechanics and Engineering*, 431:117322, 2024." The dissertation author is the primary investigator and author of this paper.

A portion of this chapter is currently being prepared for submission for publication in "H. Zhao, J. T. Hwang, and J. S. Chen. Open-source shape optimization for isogeometric shells using FEniCS and OpenMDAO". The dissertation author was the primary investigator of this material.

Chapter 3

Non-matching coupling of isogeometric shells

Structural analysis is crucial for the evaluation of the structural performance and sensitivity calculation for shape optimization. In this work, shell structures are modeled using the Kirchhoff–Love shell theory discretized by NURBS basis functions with higher order continuity. Under this framework, separate shell patches in the CAD geometry are coupled using a penalty-based formulation.

3.1 Kirchhoff–Love shell theory

This section only provides an overview to lay the foundation for the subsequent optimization approach. In the Kirchhoff–Love shell theory [103], the 3D shell continuum is represented by its mid-surface, which can be parametrized in a 2D space using coordinates $\boldsymbol{\xi} = \{\xi_1, \xi_2\}$. We denote the geometry of the mid-surface in the reference configuration as $\mathbf{X}(\boldsymbol{\xi})$ and the deformed configuration as $\mathbf{x}(\boldsymbol{\xi})$. The displacement field of the mid-surface is given by

$$\mathbf{x}(\boldsymbol{\xi}) = \mathbf{X}(\boldsymbol{\xi}) + \mathbf{u}(\boldsymbol{\xi}) \,. \tag{3.1}$$

Covariant basis vectors of the mid-surface are defined as

$$\mathbf{A}_\alpha = \mathbf{X}_{,\xi_\alpha} \qquad \text{and} \qquad \mathbf{a}_\alpha = \mathbf{x}_{,\xi_\alpha} \ , \tag{3.2}$$

where $(\cdot)_{,\xi_\alpha} = \frac{\partial(\cdot)}{\partial \xi_\alpha}$ and $\alpha = \{1,2\}$. Unit vectors that are normal to the mid-surface are given by

$$\mathbf{A}_3 = \frac{\mathbf{A}_1 \times \mathbf{A}_2}{\|\mathbf{A}_1 \times \mathbf{A}_2\|} \qquad \text{and} \qquad \mathbf{a}_3 = \frac{\mathbf{a}_1 \times \mathbf{a}_2}{\|\mathbf{a}_1 \times \mathbf{a}_2\|} \ , \tag{3.3}$$

where $\|\cdot\|$ is the L_2 norm. With surface basis vectors in (3.2), metric coefficients in both configurations are defined as

$$A_{\alpha\beta} = \mathbf{A}_\alpha \cdot \mathbf{A}_\beta \qquad \text{and} \qquad a_{\alpha\beta} = \mathbf{a}_\alpha \cdot \mathbf{a}_\beta \ , \tag{3.4}$$

for $\alpha, \beta = \{1,2\}$, and curvature coefficients read as

$$B_{\alpha\beta} = \mathbf{A}_{\alpha,\xi_\beta} \cdot \mathbf{A}_3 = -\mathbf{A}_\alpha \cdot \mathbf{A}_{3,\xi_\beta} \qquad \text{and} \qquad b_{\alpha\beta} = \mathbf{a}_{\alpha,\xi_\beta} \cdot \mathbf{a}_3 = -\mathbf{a}_\alpha \cdot \mathbf{a}_{3,\xi_\beta} \ . \tag{3.5}$$

The membrane strain tensor and curvature change tensor coefficients are formulated as

$$\varepsilon_{\alpha\beta} = \frac{1}{2}(a_{\alpha\beta} - A_{\alpha\beta}) \qquad \text{and} \qquad \kappa_{\alpha\beta} = B_{\alpha\beta} - b_{\alpha\beta} \ . \tag{3.6}$$

We employ the St. Venant–Kirchhoff material model in this paper with material tensor $\mathbf{C}$ to express normal forces and bending moments

$$\mathbf{n} = t\,\mathbf{C} : \boldsymbol{\varepsilon} \qquad \text{and} \qquad \mathbf{m} = \frac{t^3}{12}\,\mathbf{C} : \kappa \ . \tag{3.7}$$

Using membrane strains and changes in curvature defined in (3.6) and associated force resultants in (3.7), the virtual work of the Kirchhoff–Love shell reads as

$$\delta W_{\mathrm{s}} = \delta W_{\mathrm{s}}^{\mathrm{int}} - \delta W_{\mathrm{s}}^{\mathrm{ext}} = \int_S \delta\boldsymbol{\varepsilon} : \mathbf{n} + \delta\boldsymbol{\kappa} : \mathbf{m}\,\mathrm{d}S - \int_S \delta\mathbf{u}\cdot\mathbf{f}\,\mathrm{d}S\,, \tag{3.8}$$

where S is the shell mid-surface and $\mathbf{f}$ is the external force acting on S, and $\delta W_{\mathrm{s}}^{\mathrm{int}}$ and $\delta W_{\mathrm{s}}^{\mathrm{ext}}$ represent the internal and external virtual work, respectively[1]. A detailed derivation is presented in [103, Section 3].

It is noted that the curvature coefficients in (3.5) involve second-order derivatives of the displacements $\mathbf{u}$ and mid-surface geometry $\mathbf{X}$, therefore basis functions with at least C^1 continuity on element boundaries is required. Discretization using NURBS basis functions automatically meets this requirement without additional treatment.

3.2 Penalty-based non-matching shell coupling

Many complex shell structures comprise more than one NURBS patch. A coupling approach is needed for a collection of isogeometrically discretized shell patches to make them directly available for analysis. A penalty-based coupling formulation proposed by Herrema et al. [77] is employed in our current framework. The penalty energy preserves both displacement and rotational continuities on the intersection $\mathscr{L}$ between shell patch S^{A} and S^{B}, as depicted in Figure 3.1. The virtual work of the penalty energy is given by

$$\begin{aligned}\delta W_{\mathrm{pen}}^{\mathrm{AB}} = &\int_{\mathscr{L}} \alpha_{\mathrm{d}}\,(\mathbf{u}^{\mathrm{A}} - \mathbf{u}^{\mathrm{B}})\cdot(\delta\mathbf{u}^{\mathrm{A}} - \delta\mathbf{u}^{\mathrm{B}})\,d\mathscr{L} \\ &+ \int_{\mathscr{L}} \alpha_{\mathrm{r}}\,\big((\mathbf{a}_3^{\mathrm{A}}\cdot\mathbf{a}_3^{\mathrm{B}} - \mathbf{A}_3^{\mathrm{A}}\cdot\mathbf{A}_3^{\mathrm{B}})(\delta\mathbf{a}_3^{\mathrm{A}}\cdot\delta\mathbf{a}_3^{\mathrm{B}} - \delta\mathbf{A}_3^{\mathrm{A}}\cdot\delta\mathbf{A}_3^{\mathrm{B}}) \\ &\qquad + (\mathbf{a}_n^{\mathrm{A}}\cdot\mathbf{a}_3^{\mathrm{B}} - \mathbf{A}_n^{\mathrm{A}}\cdot\mathbf{A}_3^{\mathrm{B}})(\delta\mathbf{a}_n^{\mathrm{A}}\cdot\delta\mathbf{a}_3^{\mathrm{B}} - \delta\mathbf{A}_n^{\mathrm{A}}\cdot\delta\mathbf{A}_3^{\mathrm{B}})\big)\,\mathrm{d}\mathscr{L}\,,\end{aligned} \tag{3.9}$$

[1] We use subscript "s" in symbols such as $\delta W_{\mathrm{s}}^{\mathrm{int}}$ and $\delta W_{\mathrm{s}}^{\mathrm{ext}}$ to denote the quantities on shell patches.

where $\mathbf{a}_3$ and $\mathbf{a}_n$ are normal and conormal vectors on the deformed configuration, while their counterparts in the reference configuration are denoted with uppercase letters. Computation of $\mathbf{a}_n$ is discussed in detail in Section 3.3. The scalar values α_d and α_r are penalty parameters for displacement and rotational continuities. These two parameters are constructed to account for material and geometric properties and are scaled by a problem-independent and dimensionless penalty coefficient α

$$\alpha_{\mathrm{d}} = \alpha \frac{Et}{h(1-\nu^2)} \qquad \text{and} \qquad \alpha_{\mathrm{r}} = \alpha \frac{Et^3}{12h(1-\nu^3)}\,, \tag{3.10}$$

where E, ν, and t are Young's modulus, Poisson's ratio, and shell thickness, respectively. The average element length of shell patches S^{A} and S^{B} is denoted by h. On nonuniform and anisotropic meshes, the element size "h" can be ambiguous. In this work, we take $h = \frac{1}{2}\left(h_X^{\mathrm{A}} + h_X^{\mathrm{B}}\right)$ with

$$h_X = h_\xi \sqrt{\mathrm{tr}\left(\frac{\partial \mathbf{X}}{\partial \boldsymbol{\xi}} \cdot \frac{\partial \mathbf{X}}{\partial \boldsymbol{\xi}}^T\right)}\,, \tag{3.11}$$

where h_ξ is element diameter in the spline surface parameter space with coordinates $\boldsymbol{\xi}$, and $\mathbf{X}$ are Cartesian physical-space coordinates in the shell's reference configuration.[2] The reference [77] reports averaging "the lengths of the local elements in the direction most parallel to the penalty curve" [77, page 818] to obtain "h", but we prefer the isotropic definition (3.11) to accommodate arbitrary surface–surface intersections. In any case, asymptotic behavior with respect to uniform h-refinement is the same, and numerical results from both [77] and the present study indicate that results are insensitive to moderate $O(1)$ constant factors in the penalty parameters. In particular, sensitivity with respect to the coefficient α will be explored through numerical experiments in Section 3.4, producing results similar to those of [77], despite the minor differences in formulation and entirely disjoint software implementations.

Details of the penalty formulation and coupling for composite shell structures can be

[2]For technical reasons, (3.11) is projected (in a lumped-mass L^2 sense) onto a piecewise-linear finite element space on each patch before being evaluated on the intersection curve.

found in [77, Section 2], where a wide range of effective penalty coefficients α was proposed. In this paper, we use $\alpha = 1000$ for all numerical examples, as recommended and tested in [77, 181].

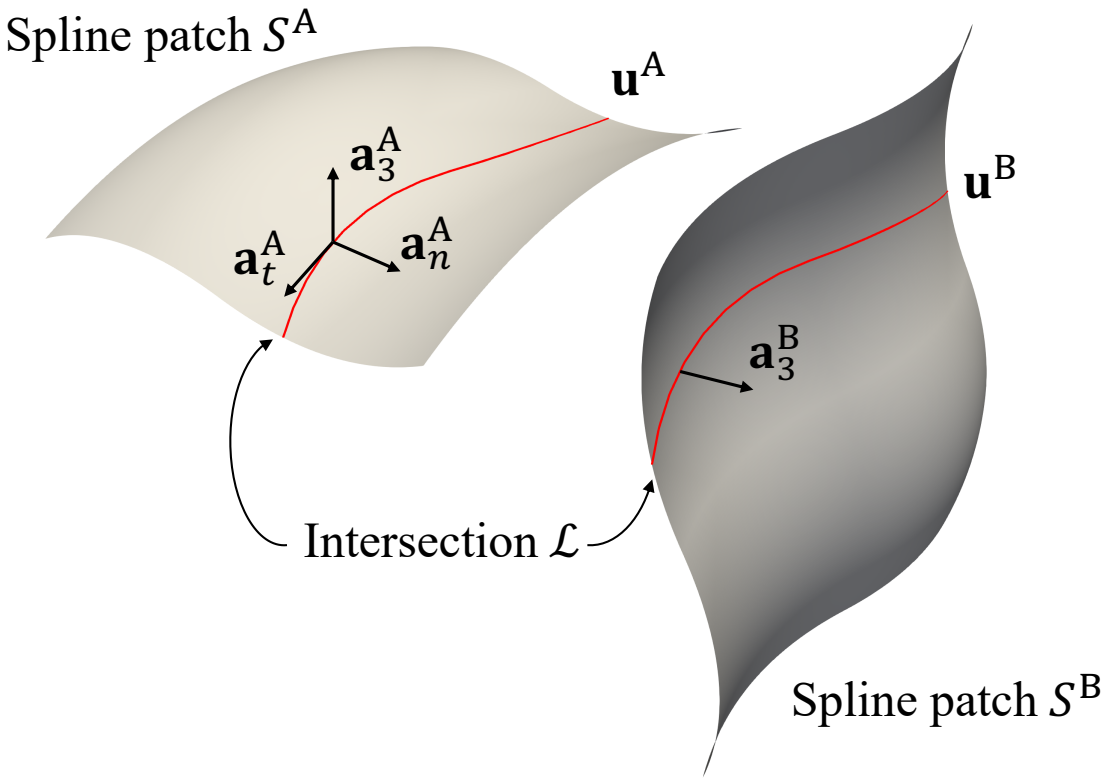


Figure 3.1. Spline patches S^{A} and S^{B} with one intersection $\mathscr{L}$ (indicated with red curves), where $\mathbf{u}$ and $\mathbf{a}_3$ are displacement and unit normal vector of midsurface, $\mathbf{a}_t$ and $\mathbf{a}_n$ are unit tangent vector and unit conormal vector on the intersection.

With the virtual work of the shell patch in (3.8) and penalty energy in (3.9), the total virtual work of two coupled shell patches S^{A} and S^{B} in the equilibrium state is expressed as

$$\delta W = \delta W_{\mathrm{s}}^{\mathrm{A}} + \delta W_{\mathrm{s}}^{\mathrm{B}} + \delta W_{\mathrm{pen}}^{\mathrm{AB}} = 0\,. \tag{3.12}$$

3.3 Numerical procedures for shell coupling

With the NURBS basis functions, Kirchhoff–Love shell geometry and displacement field are discretized isogeometrically. The position vector on the mid-surface of the shell patch in the reference configuration and the associated displacement vector are formulated as

$$\mathbf{X}(\boldsymbol{\xi}) = \sum_{i=1}^{n} N_{ip}(\boldsymbol{\xi})\mathbf{P}_i = \mathbf{N}(\boldsymbol{\xi})\mathbf{P} \qquad \text{and} \qquad \mathbf{u}(\boldsymbol{\xi}) = \sum_{i=1}^{n} N_{ip}(\boldsymbol{\xi})\mathbf{d}_i = \mathbf{N}(\boldsymbol{\xi})\mathbf{d}\,, \tag{3.13}$$

where

$$\mathbf{N}(\boldsymbol{\xi}) = \begin{bmatrix} \mathbf{I}^{sd}N_{1p}(\boldsymbol{\xi}) & \mathbf{I}^{sd}N_{2p}(\boldsymbol{\xi}) & \dots & \mathbf{I}^{sd}N_{np}(\boldsymbol{\xi}) \end{bmatrix} \tag{3.14}$$

is the matrix of NURBS basis function with degree p, and n is the number of control points, $\mathbf{I}^{sd}$ is the identity matrix in $\mathbb{R}^{sd}$ with sd as the spatial dimension. We neglect NURBS degree p in the matrix notation for conciseness. The parametric coordinate $\boldsymbol{\xi} \in \mathbb{R}^{pd}$, where pd is the parametric dimension. For the isogeometric Kirchhoff–Love shell, $sd = 3$ and $pd = 2$. $\mathbf{P}_i$ and $\mathbf{d}_i$ are vectors of mid-surface geometry control points and displacements associated with node i. Accordingly, the position vector on the mid-surface shell patch in the deformed configuration given by (3.1) is

$$\mathbf{x}(\boldsymbol{\xi}) = \mathbf{X}(\boldsymbol{\xi}) + \mathbf{u}(\boldsymbol{\xi}) = \mathbf{N}(\boldsymbol{\xi})(\mathbf{P} + \mathbf{d})\,. \tag{3.15}$$

Substituting (3.13) and (3.15) into (3.2) and following the procedures (3.4) – (3.8), we can assemble the residual force vector by taking the first derivative of the internal work (3.8) and the stiffness matrix for the second derivative[3], respectively,

[3]We use $\mathrm{d}_{\mathbf{v}}(\cdot)$ and $\partial_{\mathbf{v}}(\cdot)$ to denote the total derivative and partial derivative, respectively, of a function with respect to the discrete variables $\mathbf{v}$. This notation distinguishes from the functional derivative in the continuous setting, denoted as $(\cdot)_{,v}$, to avoid confusion.

$$\mathbf{R}_{\mathrm{s}} = \partial_{\mathbf{d}} W_{\mathrm{s}} \qquad \text{and} \qquad \mathbf{K}_{\mathrm{s}} = \partial_{\mathbf{d}} \mathbf{R}_{\mathrm{s}} \; . \tag{3.16}$$

For shell structures with single patch NURBS surface, the displacement increments can be solved by $\mathbf{K}_{\mathrm{s}} \Delta \mathbf{d} = -\mathbf{R}_{\mathrm{s}}$.

For multi-patch shell structures, contributions of the coupling term outlined in (3.9) to both membrane and bending stiffness need to be taken into consideration. Using a shell structure with two patches as an example, depicted in Figure 3.2, a topologically 1D, geometrically 2D quadrature mesh $\tilde{\Omega}$ [4] is constructed in the parameter space to represent the integration domain of the patch intersection. We first move the quadrature mesh to the parametric location of the intersection relative to shell patch S^{A}. The reference geometry and displacements of the patch intersection are obtained by interpolating corresponding functions from S^{A} to $\tilde{\Omega}$,

$$\tilde{\mathbf{X}}^{\mathrm{A}}(\zeta) = \tilde{\mathbf{N}}(\zeta)\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}})\mathbf{P}^{\mathrm{A}} = \tilde{\mathbf{N}}(\zeta)\tilde{\mathbf{P}}^{\mathrm{A}} \qquad \text{and} \qquad \tilde{\mathbf{u}}^{\mathrm{A}}(\zeta) = \tilde{\mathbf{N}}(\zeta)\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}})\mathbf{d}^{\mathrm{A}} = \tilde{\mathbf{N}}(\zeta)\tilde{\mathbf{d}}^{\mathrm{A}} \; , \tag{3.17}$$

where

$$\tilde{\mathbf{N}}(\zeta) = \begin{bmatrix} \mathbf{I}^{sd}\tilde{N}_1(\zeta) & \mathbf{I}^{sd}\tilde{N}_2(\zeta) & \dots & \mathbf{I}^{sd}\tilde{N}_m(\zeta) \end{bmatrix} \tag{3.18}$$

denotes the basis function of the quadrature mesh to approximate quantities in the physical space. Standard liner basis functions are employed for $\tilde{\mathbf{N}}(\zeta)$ in this paper, and m is the number of nodes of the quadrature mesh. $\tilde{\boldsymbol{\xi}}^{\mathrm{A}} \in \mathbb{R}^{m \cdot pd}$ refers to the vector of nodal coordinates of the quadrature mesh relative to shell patch S^{A} with $\tilde{\boldsymbol{\xi}}_i^{\mathrm{A}} \in \mathbb{R}^{pd}$. The calculation of $\tilde{\boldsymbol{\xi}}^{\mathrm{A}}$ is discussed in Section 5.1.3. Additionally, ζ is the isoparametric coordinate of the quadrature mesh, with $\zeta \in \mathbb{R}^1$ due to $\tilde{\Omega}$ being a topologically 1D mesh. $\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}) \in \mathbb{R}^{(m \cdot sd) \times (n \cdot sd)}$ is the interpolation matrix, each

[4] In this paper, all symbols indicated with ˜ denote quantities defined on the quadrature mesh of patch intersections.

row is the evaluation of the NURBS basis function of shell S^{A} at $\tilde{\boldsymbol{\xi}}_i^{\mathrm{A}}$. $\tilde{\mathbf{P}}^{\mathrm{A}}$ and $\tilde{\mathbf{d}}^{\mathrm{A}}$ are vectors of interpolated control points and displacements on the intersection. Substituting (3.17) into (3.1) and (3.2), covariant basis vectors of the mid-surface on the intersection $\mathscr{L}$ are obtained as

$$\begin{aligned}
\tilde{\mathbf{A}}_\alpha^{\mathrm{A}} &= \tilde{\mathbf{X}}^{\mathrm{A}}{}_{,\xi_\alpha} = \tilde{\mathbf{N}}(\zeta)\mathbf{N}^{\mathrm{A}}{}_{,\xi_\alpha}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}})\mathbf{P}^{\mathrm{A}} = \tilde{\mathbf{N}}(\zeta)\tilde{\mathbf{P}}_{\xi_\alpha}^{\mathrm{A}} \qquad \text{and} \\
\tilde{\mathbf{a}}_\alpha^{\mathrm{A}} &= \tilde{\mathbf{x}}^{\mathrm{A}}{}_{,\xi_\alpha} = \tilde{\mathbf{N}}(\zeta)\mathbf{N}^{\mathrm{A}}{}_{,\xi_\alpha}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}})(\mathbf{P}^{\mathrm{A}} + \mathbf{d}^{\mathrm{A}}) = \tilde{\mathbf{N}}(\zeta)(\tilde{\mathbf{P}}_{\xi_\alpha}^{\mathrm{A}} + \tilde{\mathbf{d}}_{\xi_\alpha}^{\mathrm{A}}) \,,
\end{aligned} \tag{3.19}$$

where $\mathbf{N}^{\mathrm{A}}{}_{,\xi_\alpha}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}})$ is the first order derivative of the interpolation matrix along parametric direction ξ_α, and $\tilde{\mathbf{P}}_{\xi_\alpha}^{\mathrm{A}}$ and $\tilde{\mathbf{d}}_{\xi_\alpha}^{\mathrm{A}}$ are interpolated first order derivative of the control points and displacement functions with respect to the parametric coordinates $\tilde{\boldsymbol{\xi}}^{\mathrm{A}}$ of intersection $\mathscr{L}$. Plugging (3.19) into (3.3), normal vectors of the intersection on shell S^{A} in the reference and deformed configurations can be computed as $\tilde{\mathbf{A}}_3^{\mathrm{A}}$ and $\tilde{\mathbf{a}}_3^{\mathrm{A}}$. It is notable that (3.19) requires the first order derivatives of the NURBS basis functions, ensuring rotational continuity is preserved at patch intersections.

Tangent vectors of the intersection on both configurations have to be computed before acquiring conormal vectors in (3.9), and they are given by

$$\tilde{\mathbf{A}}_t^{\mathrm{A}} = \tilde{\mathbf{X}}^{\mathrm{A}}{}_{,\zeta} = \tilde{\mathbf{N}}_{,\zeta}(\zeta)\tilde{\mathbf{P}}^{\mathrm{A}} \qquad \text{and} \qquad \tilde{\mathbf{a}}_t^{\mathrm{A}} = \tilde{\mathbf{x}}^{\mathrm{A}}{}_{,\zeta} = \tilde{\mathbf{N}}_{,\zeta}(\zeta)(\tilde{\mathbf{P}}^{\mathrm{A}} + \tilde{\mathbf{u}}^{\mathrm{A}}) \,. \tag{3.20}$$

Subsequently, conormal vectors on reference and deformed configurations are defined as

$$\tilde{\mathbf{A}}_n^{\mathrm{A}} = \frac{\tilde{\mathbf{A}}_t^{\mathrm{A}} \times \tilde{\mathbf{A}}_3^{\mathrm{A}}}{\|\tilde{\mathbf{A}}_t^{\mathrm{A}} \times \tilde{\mathbf{A}}_3^{\mathrm{A}}\|} \qquad \text{and} \qquad \tilde{\mathbf{a}}_n^{\mathrm{A}} = \frac{\tilde{\mathbf{a}}_t^{\mathrm{A}} \times \tilde{\mathbf{a}}_3^{\mathrm{A}}}{\|\tilde{\mathbf{a}}_t^{\mathrm{A}} \times \tilde{\mathbf{a}}_3^{\mathrm{A}}\|} \,. \tag{3.21}$$

Next, we move the quadrature mesh $\tilde{\Omega}$ to the parametric position defined by coordinates $\tilde{\boldsymbol{\xi}}^{\mathrm{B}}$ relative to shell patch S^{B}, where the calculation of $\tilde{\boldsymbol{\xi}}^{\mathrm{B}}$ is discussed in Section 5.1.3. By repeating (3.17) and (3.19), we can determine the displacements $\tilde{\mathbf{u}}^{\mathrm{B}}$ and normal vectors $\tilde{\mathbf{A}}_3^{\mathrm{B}}$ and $\tilde{\mathbf{a}}_3^{\mathrm{B}}$ of S^{B} at the intersection $\mathscr{L}$. Substituting these displacements and geometry vectors from the

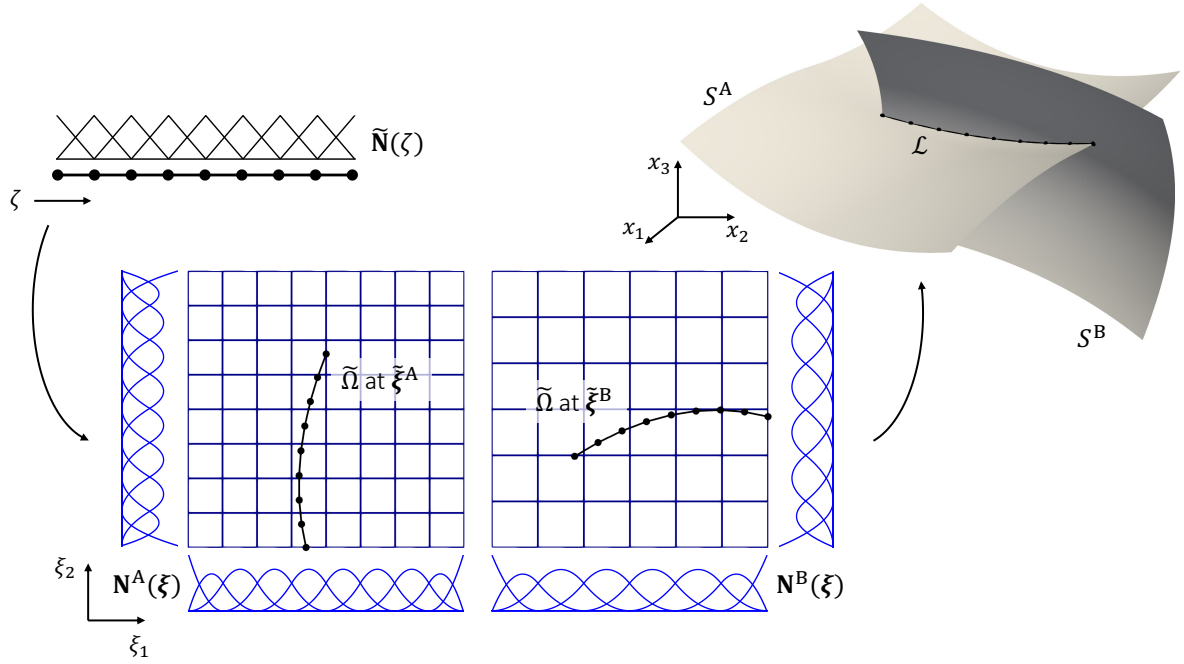


Figure 3.2. An illustrative example of two shell patches with one intersection. Shell patches S^{A} and S^{B} are discretized isogeometrically using NURBS basis functions $\mathbf{N}^{\mathrm{A}}(\boldsymbol{\xi})$ and $\mathbf{N}^{\mathrm{B}}(\boldsymbol{\xi})$. A topologically 1D quadrature mesh $\tilde{\Omega}$, discretized using linear basis functions $\tilde{\mathbf{N}}(\zeta)$, is created in the parametric space to integrate the penalty energy for shell coupling.

quadrature mesh into (3.9), the penalty virtual work $\delta W_{\mathrm{pen}}^{\mathrm{AB}}(\tilde{\mathbf{d}}, \tilde{\mathbf{d}}_{\boldsymbol{\xi}}, \tilde{\mathbf{P}}, \tilde{\mathbf{P}}_{\boldsymbol{\xi}})$ can be integrated on $\tilde{\Omega}$, where $\tilde{\mathbf{d}}$ and $\tilde{\mathbf{P}}$ are the interpolated displacements and geometric control points for both surfaces. $\tilde{\mathbf{d}}_{\boldsymbol{\xi}}$ and $\tilde{\mathbf{P}}_{\boldsymbol{\xi}}$ are the associated first order derivatives. Consequently, the residual force vector and stiffness matrix of the coupled shell structure are

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{\mathrm{s}}^{\mathrm{A}} + \mathbf{R}_{\mathrm{pen}}^{\mathrm{A}} \\ \mathbf{R}_{\mathrm{s}}^{\mathrm{B}} + \mathbf{R}_{\mathrm{pen}}^{\mathrm{B}} \end{bmatrix} \quad \text{and} \quad \mathbf{K} = \begin{bmatrix} \mathbf{K}_{\mathrm{s}}^{\mathrm{A}} + \mathbf{K}_{\mathrm{pen}}^{\mathrm{AA}} & \mathbf{K}_{\mathrm{pen}}^{\mathrm{AB}} \\ \mathbf{K}_{\mathrm{pen}}^{\mathrm{BA}} & \mathbf{K}_{\mathrm{s}}^{\mathrm{B}} + \mathbf{K}_{\mathrm{pen}}^{\mathrm{BB}} \end{bmatrix}, \tag{3.22}$$

where components of penalty energy contribution, e.g., $\mathbf{R}_{\mathrm{pen}}^{\mathrm{A}}$ and $\mathbf{K}_{\mathrm{pen}}^{\mathrm{AB}}$, are defined as

$$\mathbf{R}_{\mathrm{pen}}^{\mathrm{A}} = (\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}))^{\mathrm{T}} \partial_{\tilde{\mathbf{d}}^{\mathrm{A}}} W_{\mathrm{pen}}^{\mathrm{AB}} + (\mathbf{N}_{,\boldsymbol{\xi}}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}))^{\mathrm{T}} \partial_{\tilde{\mathbf{d}}_{\boldsymbol{\xi}}^{\mathrm{A}}} W_{\mathrm{pen}}^{\mathrm{AB}} \quad \text{and} \tag{3.23}$$

$$\mathbf{K}_{\mathrm{pen}}^{\mathrm{AB}} = (\mathbf{N}^{\mathrm{B}}(\tilde{\boldsymbol{\xi}}^{\mathrm{B}}))^{\mathrm{T}} \partial_{\tilde{\mathbf{d}}^{\mathrm{B}}} \mathbf{R}_{\mathrm{pen}}^{\mathrm{A}} + (\mathbf{N}_{,\boldsymbol{\xi}}^{\mathrm{B}}(\tilde{\boldsymbol{\xi}}^{\mathrm{B}}))^{\mathrm{T}} \partial_{\tilde{\mathbf{d}}_{\boldsymbol{\xi}}^{\mathrm{B}}} \mathbf{R}_{\mathrm{pen}}^{\mathrm{A}} . \tag{3.24}$$

And $\mathbf{N}_{,\boldsymbol{\xi}}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}) \in \mathbb{R}^{(m\cdot pd\cdot sd)\times(n\cdot sd)}$ is the first order derivative of the interpolation matrix on both parametric directions.

The displacement increments for both spline patches can be solved using the Newton–Raphson method, as expressed by $\mathbf{K}\Delta\mathbf{d} = -\mathbf{R}$. Equation (3.24) indicates that $\mathbf{K}_{\mathrm{pen}}^{\mathrm{AB}} = \mathbf{K}_{\mathrm{pen}}^{\mathrm{BA}^{\mathrm{T}}}$, enabling the lower triangle blocks in $\mathbf{K}$ to be obtained from the upper triangle counterparts, thereby improving computational efficiency. Readers are referred to [181] for details about implementation and code framework. A series of benchmark problems in [181, Section 4] have been utilized to verify the accuracy of this method, which shows good results for the application of aircraft wings [180, Section 6].

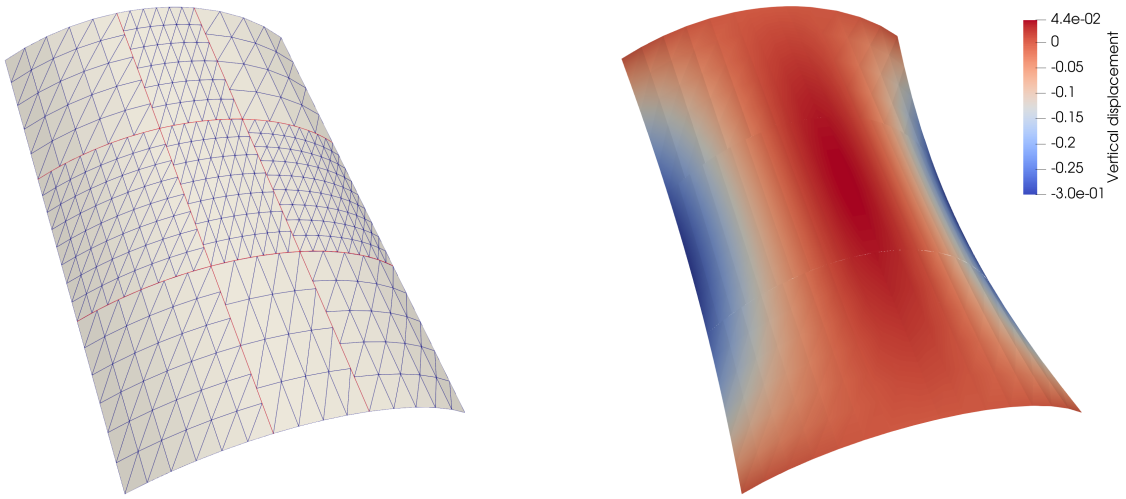
3.4 Benchmark problems

This section demonstrates the use of penalty-based coupling formulation to solve several benchmark problems from the literature. Each problem is selected to verify specific functionality from PENGoLINS. Unless otherwise specified, results are obtained using the default value of $\alpha = 10^3$ for the dimensionless penalty coefficient.

3.4.1 Scordelis-Lo roof

The first benchmark we consider is the Scordelis–Lo roof example [16], where we divide its geometry into nine separately-parameterized NURBS surfaces, as depicted in Figure 3.3a. In this benchmark, the cylindrical roof is subjected to self-weight while constraints are applied to the two curved edges. Full details of the geometry, boundary conditions, material properties, etc. are provided in [77, Section 3.1]. The computed displacement in the vertical direction for geometry in Figure 3.3a is shown in Figure 3.3b. This plot clearly indicates that our framework is able to maintain approximate displacement continuity on the non-matching interfaces during deformation.

To verify the numerical solution, the convergence of a quantity of interest (QoI), viz., vertical displacement at the midpoint of a free edge, is plotted for quadratic, cubic, and quartic



(a) **(b)**

Figure 3.3. (a) Scordelis–Lo roof geometry, consisting of nine non-matching NURBS surfaces, with a total of 2259 DoFs. The twelve non-matching interfaces are indicated with red color. (b) Vertical displacement of the Scordelis–Lo roof, using a scale factor of 10 to warp the initial geometry. Results are interpolated onto piecewise linear triangle elements for visualization purposes.

NURBS surfaces in Figure 3.4. The converged solution of the QoI is reported in [103, Section 6.2.1] as 0.3006. All three degrees of NURBS surface converge to the reference value when refined uniformly via knot insertion. The representative solution plotted in Figure 3.3b corresponds to the first data point of cubic NURBS surfaces in Figure 3.4.

To investigate the sensitivity of results with respect to the dimensionless penalty coefficient, we plot the QoI as a function of penalty coefficient (for cubic NURBS) in Figure 3.5. The results indicate that a wide range of penalty coefficients can produce accurate results in the Scordelis–Lo roof example, even with relatively few DoFs. Overall, the default value of $\alpha = 10^3$ recommended by [77] works for all mesh refinements, despite slight changes to the formulation, and we see a similar overall sensitivity to α.

An important application of this work is stress analysis of shell structures. We compute the stress resultants, viz., normal forces, bending moments, and transverse shear forces, for the Scordelis–Lo roof and plot their first components in Figure 3.6. Methods for stress resultant

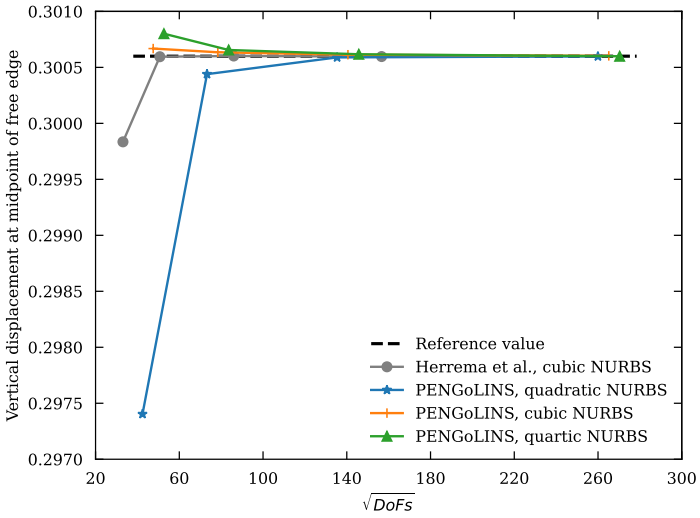


Figure 3.4. Convergence of Scordelis–Lo roof example for different orders of NURBS surfaces. Results from [77] (using a different mesh structure) are included for reference.

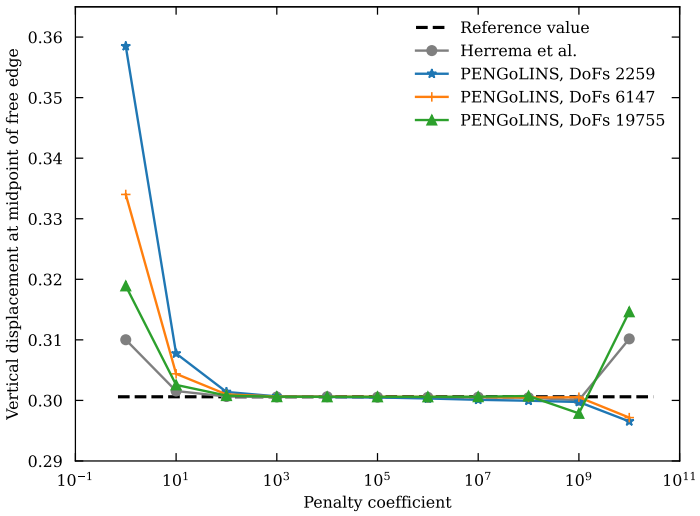


Figure 3.5. A wide range of penalty coefficients compute accurate results for Scordelis–Lo roof example. Results from [77, Figure 6(a), non-matching] are included for reference.

computation and stress recovery in isogeometric Kirchhoff–Love shells are detailed by [103, Section 3.4], and we follow the notation of the cited reference when reporting results here. Reference values for stress resultants, computed by Abaqus shell analysis, are given in [103, Section 6.2.4]. The resultant distributions shown in Figure 3.6 qualitatively agree with the reference results. Slight oscillations are visible in the transverse shear resultant, where corners

of multiple patches are joined together, but the resultant distributions are otherwise free from visible artifacts at the non-matching interfaces. A quantitative comparison of extreme values is compiled into Table 3.1, also demonstrating good agreement, especially of maximum absolute values, which are most relevant to design.

Instability in transverse shear resultants at intersections is not surprising, because these resultants are obtained by directly evaluating *third* derivatives of the displacement [103, (3.61)]. The cited formula assumes C^2 continuity of the displacement; C^2 continuity holds within patches for the cubic displacement solutions postprocessed in Figure 3.6, but the penalty formulation only (approximately) enforces C^1 continuity at intersections.

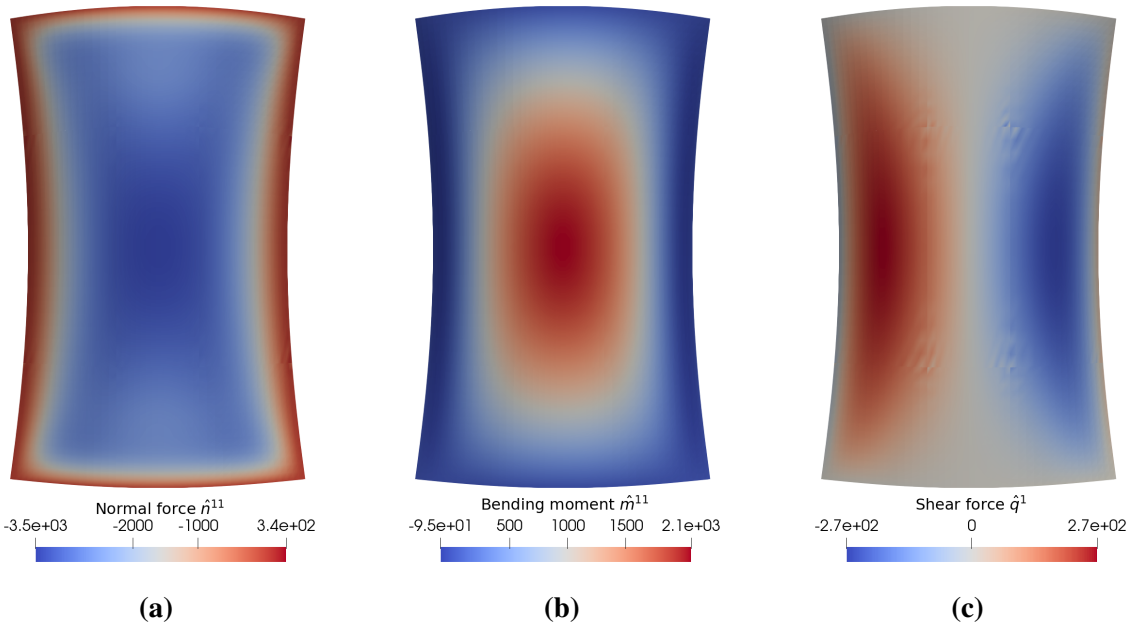


Figure 3.6. Stress resultants from the Scordelis–Lo roof with 9 non-matching shell patches and 9819 DoFs in total. (a) First component of normal forces $\hat{n}^{11}$. (b) First component of bending moments $\hat{m}^{11}$. (c) First component of shear forces $\hat{q}^1$.

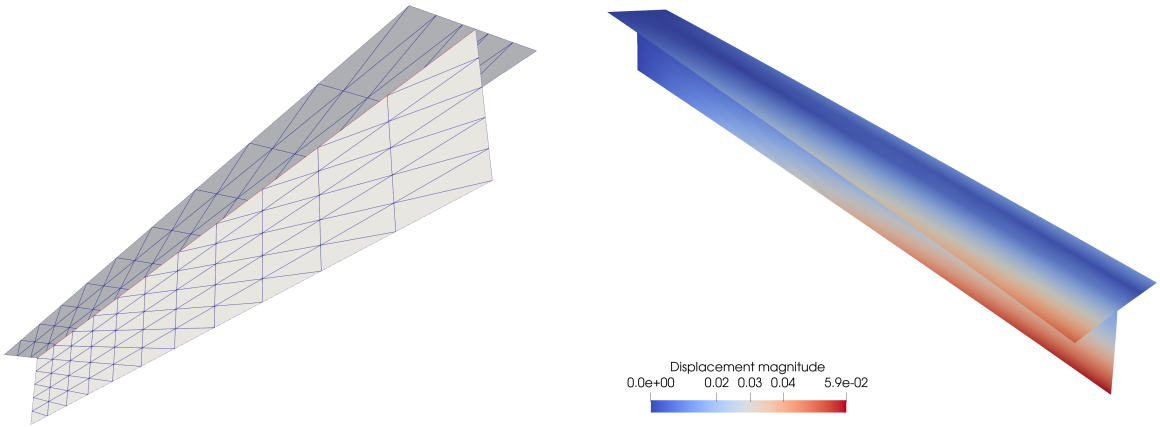
3.4.2 Torsion of a T-beam

We now consider a benchmark in which shell patches are joined together at an angle. In particular, we consider a T-beam consisting of two mismatched cubic NURBS surfaces, as

Table 3.1. Comparison of stress resultants between non-matching Kirchhoff–Love shell analysis and the Abaqus reference computation, as well as single patch analysis results reported in [103, Section 6.2.4].

Stress resultants	Non-matching shell analysis		Abaqus shell analysis		Single patch shell analysis	
	Min	Max	Min	Max	Min	Max
$\hat{n}^{11}$	-3487	336	-3417	127	-3510	25
$\hat{m}^{11}$	-95	2055	-93	2079	-91	2053
$\hat{q}^{1}$	-273	273	-278	278	-280	280

depicted in Figure 3.7a, with 648 DoFs. One end of the T-beam is fully pinned, and a downward vertical point load is applied to one corner of the opposite end. Complete dimensions, boundary conditions, loading, and material parameters for this example can be found in [77, Section 3.3]. However, unlike the problem setup in [77], our discretization uses only one patch for the top of the "T", and the junction with the vertical patch is not located at a knot of the top patch. The displacement distribution of the deformed T-beam is shown in Figure 3.7b.

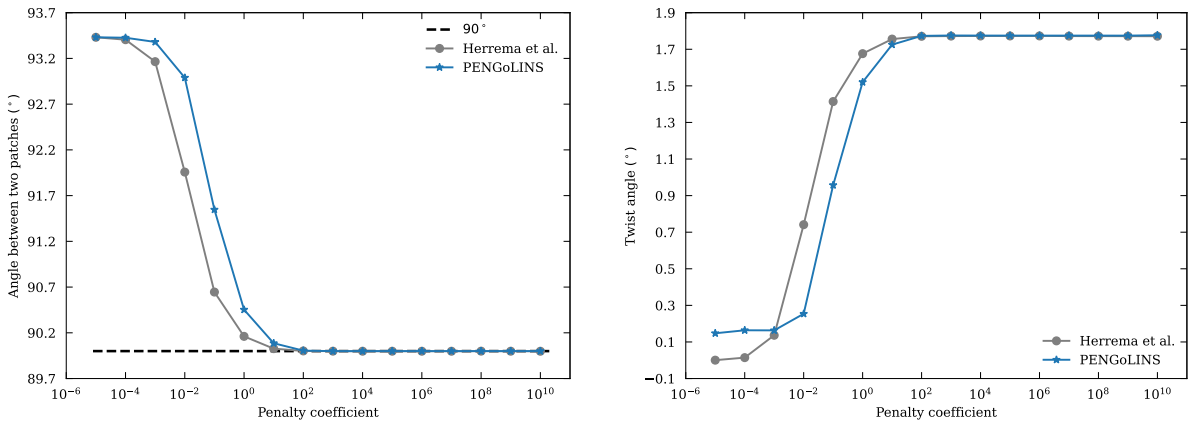


(a) **(b)**

Figure 3.7. (a) T-beam geometry consisting of two separate NURBS surfaces with 648 DoFs in total. Note that the non-matching interface does not coincide with a knot of the horizontal patch. (b) Displacement of the T-beam benchmark test, using a scale factor of 10 to warp the initial geometry.

We use this benchmark to test the formulation's ability to maintain the 90° angle between

the horizontal and vertical shell patches at the free end. In particular, we examine the sensitivity of the deformed angle to the dimensionless penalty coefficient, as illustrated in Figure 3.8a. The angle between the two patches cannot be approximated well in the deformed state for a penalty coefficient smaller than 100, but, as the value of penalty coefficient increases, the penalty terms become sufficiently strong, and the angle converges to exactly 90°, as expected. A similar convergence pattern is seen in Figure 3.8b, where the twist angle between two ends of the vertical patch converges to a specific value for penalty coefficient greater than 100. Both figures display similar effects of penalty coefficient to results in [77, Figure 15]. Thus, Figure 3.8 demonstrates that the recommended penalty coefficient of 10^3 identified in Section 3.4.1 remains effective at preserving rotational continuity when patches meet at a nonzero angle in the initial configuration. Comparing with the results of [77] for this problem, we see that the interpretation of "α" differs by roughly a constant factor (i.e., a horizontal translation when using a log scale) over most of the range considered. This is consistent with what we expect from using a different mesh and definition of "h" in the penalty parameters, but even with these differences, results are practically indistinguishable for α around or beyond the recommended value of 10^3.



(a) **(b)**

Figure 3.8. (a) The angle between the two patches of the T-beam at the free end, as a function of penalty coefficient. (b) The twist angle of the vertical patch as a function of the penalty coefficient. Results from [77] (using a different mesh and definition of element size) are included for reference.

3.4.3 Nonlinear analysis of a slit annular plate

This section considers geometrically nonlinear analysis of a slit annular plate. The plate is subjected to a vertical line force along one side of the slit, with a maximum magnitude of $P_{max} = 0.8$ per unit length. The other side of the slit is clamped. This benchmark test was first proposed by Sze et al. [157], and readers can find the full problem definition in Section 3.3 of the cited reference. The geometry is composed of four cubic NURBS surfaces, as shown in Figure 3.9a. Figure 3.9b shows the resulting deformed configuration at the maximum load. Displacement remains qualitatively smooth across the non-matching interfaces.

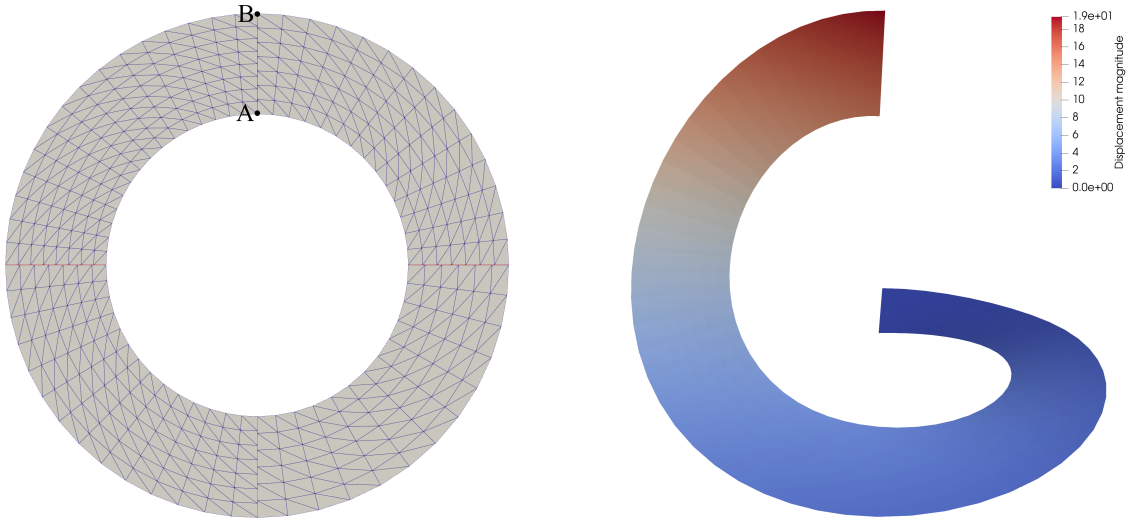


Figure 3.9. (a) Slit annular plate geometry, split into four NURBS surfaces, with a total of 2052 DoFs. The three non-matching interfaces are marked by red lines. (b) Displacement of the slit annular plate at the maximum magnitude of the applied line load.

We compare the vertical displacements of two points, labelled A and B in Figure 3.9a, with reference values provided in [157, Table 4] and benchmark results reported in [77, Figure 23]. Figure 3.10 shows that the vertical displacements at points A and B, W_A and W_B, are consistent with the reference values. The recommended value for the penalty coefficient, 10^3, is still sufficient to maintain approximate displacement and rotational continuity in this benchmark. The slit annular plate example demonstrates that our framework produces accurate results for

geometrically nonlinear problems with large rotations.

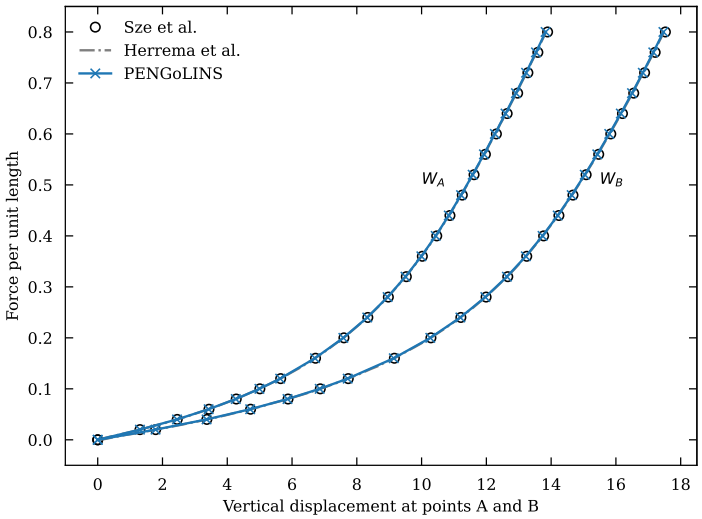


Figure 3.10. Comparison between our computations and reference data for vertical displacement at points A and B.

3.4.4 Shell structures with curved intersections

In the benchmark tests of Section 3.4.1–3.4.3, the intersection curves between patches are parallel to coordinate lines of one spline parameter and orthogonal to coordinate lines of the other. This section considers cases where the intersection curves are not aligned with the spline parameterizations. In particular, we modify the Scordelis–Lo roof and T-beam examples to have more complex parameterizations.

For the Scordelis–Lo roof, we distort the parameterization as shown in Figure 3.11, such that the intersection is no longer orthogonal (in physical space) to parametric coordinate lines running in the axial direction. Figure 3.12 compares the convergence of displacement using this parameterization with the results from Section 3.4.1. The formulation clearly still converges rapidly under h-refinement using discretizations of various polynomial degrees, albeit with moderately weaker per-DoF approximation power, especially for quadratic NURBS. For the T-beam, we distort the parameterization of the top patch as shown in Figure 3.13, such that the intersection becomes curved in the spline parameter space. Figure 3.14 demonstrates that

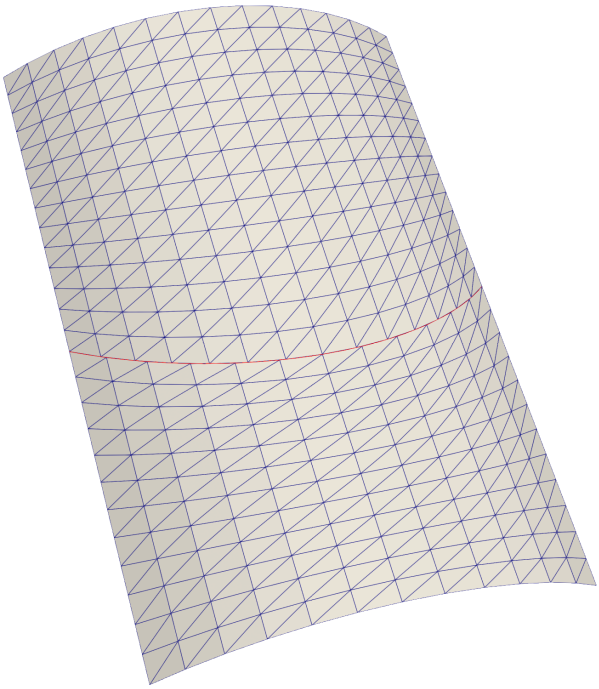


Figure 3.11. Distorted parameterization of the Scordelis–Lo roof.

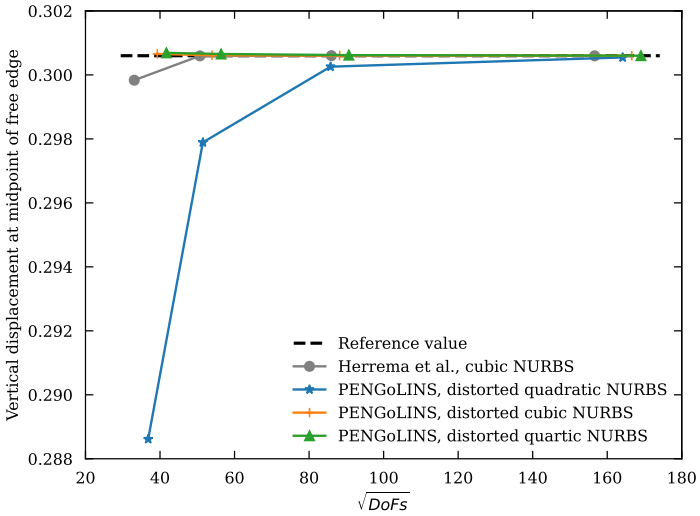


Figure 3.12. Convergence of displacement in the Scordelis–Lo roof with a distorted parameterization.

the angular constraint between the patches and the overall twist angle are still approximated accurately, with similar dependencies on α as observed when the intersection is aligned with the top patch's parameterization, as in both Figure 3.8 of this paper and [77]. We can conclude from

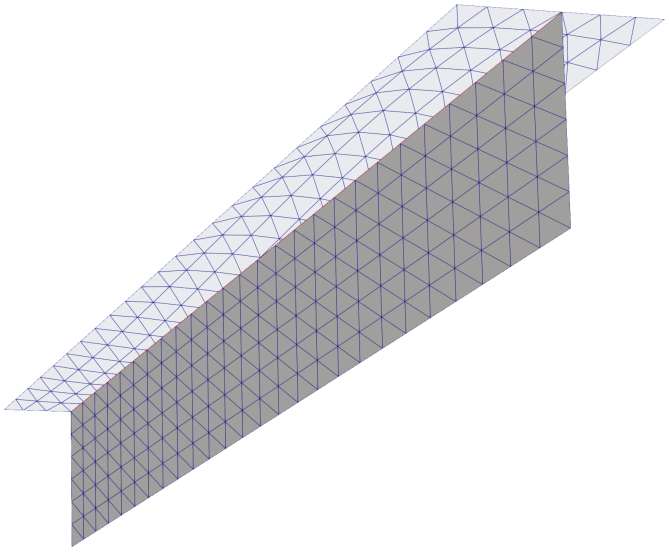
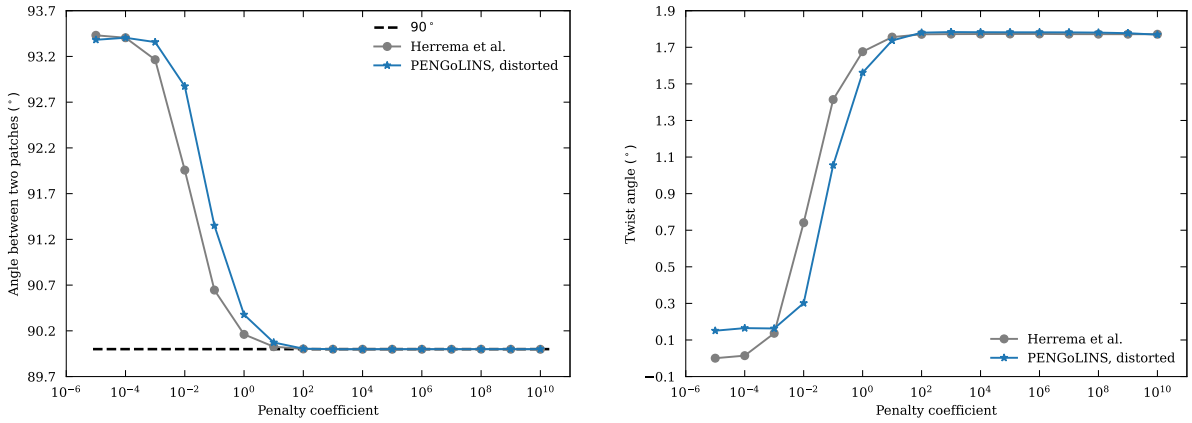


Figure 3.13. T-beam with a distorted parameterization of the top patch.



(a) **(b)**

Figure 3.14. The angle between the T-beam patches (a) and the twist angle of the vertical patch (b) as functions of the penalty coefficient α, using the distorted parameterization of Figure 3.13. We again include the results of [77] for comparison, although the interpretation of α differs by a constant factor when using a different mesh.

the additional tests of this section that the penalty formulation is robust with respect to different relative orientations of intersection curves and parametric coordinate lines.

Acknowledgements

A portion of this chapter has been published in "H. Zhao, X. Liu, A. H. Fletcher, R. Xiang, J. T. Hwang, and D. Kamensky. An open-source framework for coupling non-matching isogeometric shells with application to aerospace structures *Computers & Mathematics with Applications*, 111:109–123, 2022." The dissertation author is the primary investigator and author of this paper.

A portion of this chapter has been published in "H. Zhao, J. T. Hwang, and J. S. Chen. Shape optimization of non-matching isogeometric shells with moving intersections. *Computer Methods in Applied Mechanics and Engineering*, 431:117322, 2024." The dissertation author is the primary investigator and author of this paper.

Chapter 4

Shape optimization using FFD

CAD geometries of Kirchhoff–Love shells can be used for analysis without finite element mesh generation by employing formulations from Chapter 3. This provides attractive features for shape optimization of shell structures, where the discretization of shell patches stays unaltered during shape evolution. The updated geometry in optimization iterations stays consistent with the analysis model. As a result, this approach minimizes the effort required for geometry processing while simultaneously enhancing accuracy.

In the context of shape optimization for non-matching shell structures, it is crucial to ensure that updated shell patches remain properly connected. Failure to maintain this connectivity can result in separation or self-penetration of shell patches during the optimization iteration. Such issues would lead to false analysis and yield unrealistic optimal shapes. To tackle this challenge, we adopt the FFD-based [152] technique combined with Lagrange extraction to update shell geometry and demonstrate the workflow in Section 4.2. A comparable concept can be applied to thickness optimization to ensure continuous thickness distribution at the surface–surface intersections if needed. Sensitivities for both shape and thickness optimization are given in the subsequent sections.

4.1 Automate IGA with Lagrange extraction

The concept of extraction [23, 150, 148, 54] is utilized in the implementation of IGA, whose spline basis functions can be represented exactly by the linear combination of Lagrange basis functions. These Lagrange basis functions can be used in the classical FEM, allowing IGA to be implemented using finite element software with pre-defined extraction operators. An open-source IGA Python library named tIGAr is developed by Kamensky et al. [98] using the finite element software FEniCS [122]. Implementation and technical details are discussed in Chapter 7. In this section, we illustrate the basic mathematical operations and workflow of IGA using extraction.

To perform IGA, an extraction matrix $\mathbf{M}$ is generated to represent functions defined in spline function space $\mathscr{V}$ with FE basis functions in $\mathscr{V}^{\mathrm{FE}}$. The relation between these two sets of basis functions is given by

$$\mathbf{N} = \mathbf{M}^{\mathrm{T}}\mathbf{N}^{\mathrm{FE}} , \tag{4.1}$$

where $\mathbf{N}$ are IGA basis functions and $\mathbf{N}^{\mathrm{FE}}$ are FE basis functions. Each column of $\mathbf{M}$ is the linear combination of $\mathbf{N}^{\mathrm{FE}}$ giving an IGA basis function. In the analysis, we first create an extraction matrix $\mathbf{M}$ and assemble the stiffness matrix $\mathbf{K}^{\mathrm{FE}}$ and force vector $\mathbf{F}^{\mathrm{FE}}$ in $\mathscr{V}^{\mathrm{FE}}$ using existing finite element subroutines. Then the displacement in $\mathscr{V}$ is solved as

$$(\mathbf{M}^{\mathrm{T}}\mathbf{K}^{\mathrm{FE}}\mathbf{M})\mathbf{d} = \mathbf{M}^{\mathrm{T}}\mathbf{F}^{\mathrm{FE}} , \tag{4.2}$$

with problem-specific boundary conditions applied to $\mathbf{M}^{\mathrm{T}}\mathbf{K}^{\mathrm{FE}}\mathbf{M}$ and $\mathbf{M}^{\mathrm{T}}\mathbf{F}^{\mathrm{FE}}$.

For the purpose of clarity, we assume control points of spline surfaces have unit weights. Therefore, rational spline basis functions are the same as homogeneous spline basis functions, both denoted as $\mathbf{N}$. In practice, weights need to be taken into consideration for correct geometric mapping and analysis.

For single patch Kirchhoff–Love shell analysis, stiffness matrix $\mathbf{K}^{\mathrm{FE}}$ is the second derivative of total work $\partial_{\mathbf{d}^{\mathrm{FE}}}(\partial_{\mathbf{d}^{\mathrm{FE}}}W)$. $\mathbf{M}^{\mathrm{T}}\partial_{\mathbf{d}^{\mathrm{FE}}}(\partial_{\mathbf{d}^{\mathrm{FE}}}W)\mathbf{M}$ changes basis of $\partial_{\mathbf{d}^{\mathrm{FE}}}(\partial_{\mathbf{d}^{\mathrm{FE}}}W)$ from $\mathscr{V}^{\mathrm{FE}}$ to $\mathscr{V}$, and the formulation of IGA stiffness matrix can also be expressed as

$$\mathbf{M}^{\mathrm{T}}\partial_{\mathbf{d}^{\mathrm{FE}}}(\partial_{\mathbf{d}^{\mathrm{FE}}}W)\mathbf{M} = \left(\partial_{\mathbf{d}}\mathbf{d}^{\mathrm{FE}}\right)^{\mathrm{T}}\partial_{\mathbf{d}^{\mathrm{FE}}}(\partial_{\mathbf{d}^{\mathrm{FE}}}W)\partial_{\mathbf{d}}\mathbf{d}^{\mathrm{FE}} = \partial_{\mathbf{d}}(\partial_{\mathbf{d}}W)\ . \tag{4.3}$$

The right-hand side (RHS) of (4.2) for Kirchhoff–Love shell is equivalent to

$$\mathbf{M}^{\mathrm{T}}\left(-\partial_{\mathbf{d}^{\mathrm{FE}}}W\right) = \left(\partial_{\mathbf{d}}\mathbf{d}^{\mathrm{FE}}\right)^{\mathrm{T}}\left(-\partial_{\mathbf{d}^{\mathrm{FE}}}W\right) = -\partial_{\mathbf{d}}W\ . \tag{4.4}$$

Therefore, (4.2) is recovered as the linear system in $\mathscr{V}$ to solve for displacements increments in IGA DoFs,

$$\partial_{\mathbf{d}}(\partial_{\mathbf{d}}W)\Delta\mathbf{d} = -\partial_{\mathbf{d}}W\ . \tag{4.5}$$

While the Kirchhoff–Love shell theory neglects transverse shear strains, membrane locking can still occur in cases of curved geometry. In our current implementation, we use the standard Gauss quadrature rule to perform numerical integration without including special treatment to avoid membrane locking. We have conducted numerical studies in our earlier work [181, Section 4] to assess the accuracy and convergence of the framework for a series of benchmark problems. Solutions converge quickly to the reference solution with increased refinement, particularly evident for cubic NURBS surfaces, which are utilized in the numerical examples and applications in the following sections. Nevertheless, incorporating methods to alleviate potential membrane locking in isogeometric Kirchhoff–Love shells into our open-source framework will be an appealing research topic in the future. We refer readers who are interested in techniques for removing membrane locking to [156, 64, 19, 27, 29, 65, 134, 145].

4.2 Non-matching shells update through FFD block

We use a cylindrical roof consisting of four non-matching shell patches that are shown in the upper-left part of Figure 4.1 as an example to demonstrate the FFD-based shape optimization approach. Note that this approach can be applied to shell structures with an arbitrary number of patches.

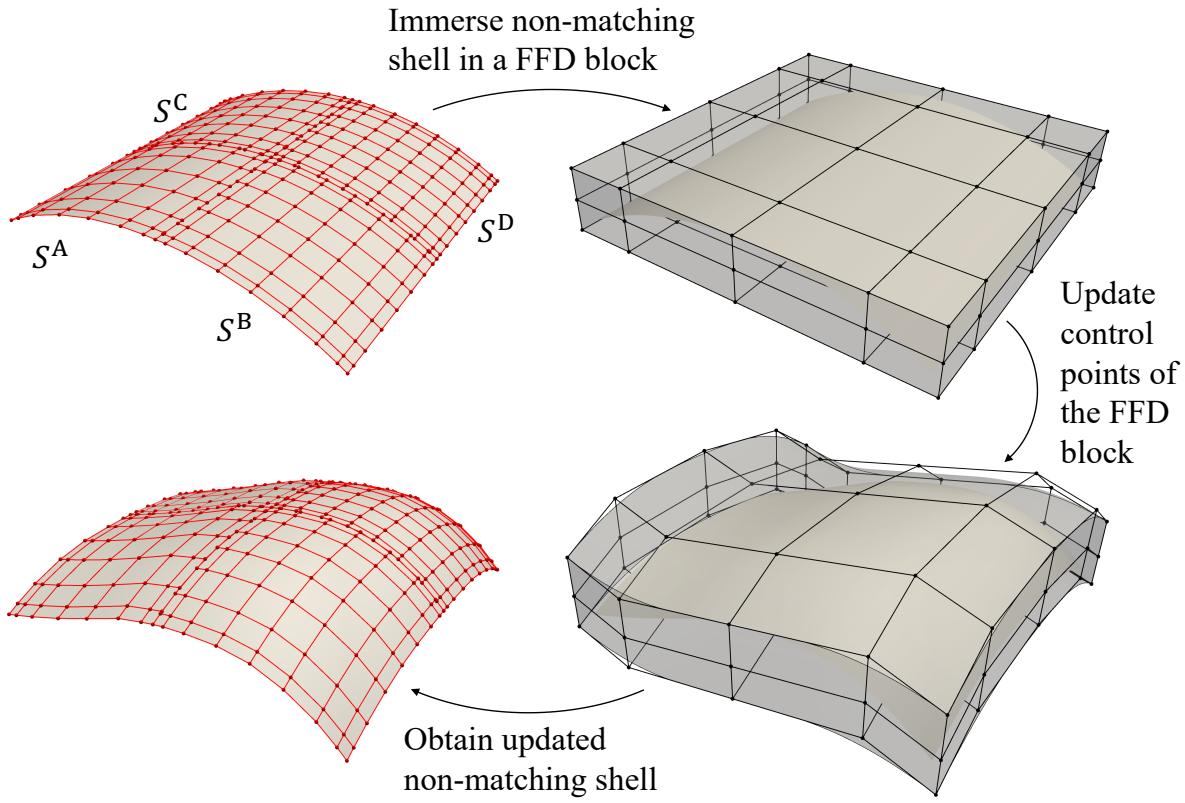


Figure 4.1. Workflow of FFD-based shape optimization for non-matching shell structures. A cylindrical roof consisting of four non-matching NURBS patches is first immersed in a trivariate B-spline block. We update the control points of the FFD block to deform the shape of the non-matching cylindrical roof. Control nets of the NURBS surfaces are indicated with red color, and black is used for the control net of the FFD block.

For the initial CAD geometry consisting of m Kirchhoff–Love shell patches, define a set of NURBS surface functions $\{\mathbf{S}^{\mathrm{A}}(\boldsymbol{\xi}),\mathbf{S}^{\mathrm{B}}(\boldsymbol{\xi}),\ldots,\mathbf{S}^{\mathrm{I}^m}(\boldsymbol{\xi})\}$, and the I-th shell patch $\mathbf{S}^{\mathrm{I}}(\boldsymbol{\xi})$ is

expressed as

$$\mathbf{S}^{\mathrm{I}}(\boldsymbol{\xi}) = \mathbf{N}^{\mathrm{I}}(\boldsymbol{\xi})\mathbf{P}^{\mathrm{I}} \,, \tag{4.6}$$

where $\mathbf{N}^{\mathrm{I}}(\boldsymbol{\xi})$ are the spline basis functions of degree p_{sh} in $\mathscr{V}^{\mathrm{I}}$. We omit degree p_{sh} in the notation for clarity. $\mathbf{P}^{\mathrm{I}}$ are the NURBS control points for surface function $\mathbf{S}^{\mathrm{I}}$.

Using the extraction concept [148], NURBS surface function $\mathbf{S}^{\mathrm{I}}(\boldsymbol{\xi})$ can be represented with Lagrange polynomials as well,

$$\mathbf{S}^{\mathrm{I}}(\boldsymbol{\xi}) = \mathbf{N}^{\mathrm{I,FE}}(\boldsymbol{\xi})\mathbf{P}^{\mathrm{I,FE}} \,, \tag{4.7}$$

where $\mathbf{N}^{\mathrm{I,FE}}(\boldsymbol{\xi})$ are basis functions in the finite element function space $\mathscr{V}_{\mathrm{s}}^{\mathrm{I,FE}}$ with nodal interpolatory property, and $\mathbf{P}^{\mathrm{I,FE}}$ are Lagrange control points, or nodal values of $\mathbf{S}^{\mathrm{I}}$. Plugging nodal coordinate $\boldsymbol{\xi}^{\mathrm{I,FE}}$ of $\mathscr{V}_{\mathrm{s}}^{\mathrm{I,FE}}$ into (4.7), coordinate of the NURBS surface $\mathbf{S}^{\mathrm{I}}$ is represented with nodal values in the discrete setting,

$$\mathbf{S}^{\mathrm{I}}(\boldsymbol{\xi}^{\mathrm{I,FE}}) = \mathbf{N}^{\mathrm{I,FE}}(\boldsymbol{\xi}^{\mathrm{I,FE}})\mathbf{P}^{\mathrm{I,FE}} = \mathbf{P}^{\mathrm{I,FE}} \,. \tag{4.8}$$

Based on (4.1), Lagrange control points can be obtained through the extraction matrix and NURBS control points. We have the following relation,

$$\mathbf{S}^{\mathrm{I}}(\boldsymbol{\xi}^{\mathrm{I,FE}}) = \mathbf{P}^{\mathrm{I,FE}} = \mathbf{M}^{\mathrm{I}}\mathbf{P}^{\mathrm{I}} \,. \tag{4.9}$$

The first step of Figure 4.1 illustrates the initial configuration of a collection of intersecting non-matching shell patches S, where red control nets are displayed. To enforce connectivity of the intersections during optimization, we immerse S in a trivariate B-spline block, which is refered to as an FFD block, and use control points of the FFD block as design variables. A schematic demonstration is shown in the second step of Figure 4.1. The FFD B-spline block is

defined as

$$\mathbf{V}(\boldsymbol{\theta}) = \mathbf{N}_{\mathrm{FFD}}(\boldsymbol{\theta})\mathbf{P}_{\mathrm{FFD}} \,, \tag{4.10}$$

where $\boldsymbol{\theta}$ is the parametric coordinate of the FFD block, $\mathbf{N}_{\mathrm{FFD}}(\boldsymbol{\theta})$ are B-spline solid basis functions of degree p_{FFD} with knots vector, and $\mathbf{P}_{\mathrm{FFD}}$ are B-spline block control points.

To simplify formulation and implementation, we use an identity mapping for the FFD block B-spline block, so that the parametric coordinate coincides with the physical coordinate,

$$\mathbf{V}(\boldsymbol{\theta}) = \mathbf{N}_{\mathrm{FFD}}(\boldsymbol{\theta})\mathbf{P}_{\mathrm{FFD}} = \boldsymbol{\theta} \,. \tag{4.11}$$

Substituting (4.9) into (4.11), NURBS surfaces of the non-matching shells can be expressed using the FFD block basis functions and control points,

$$\mathbf{V}\left(\mathbf{S}^{\mathrm{I}}(\boldsymbol{\xi})\right) = \mathbf{N}_{\mathrm{FFD}}\left(\mathbf{S}^{\mathrm{I}}(\boldsymbol{\xi})\right)\mathbf{P}_{\mathrm{FFD}} = \mathbf{S}^{\mathrm{I}}(\boldsymbol{\xi}) \,. \tag{4.12}$$

In the continuous context of (4.12), shell patches will not separate in the final configuration as long as they are interconnected in the initial geometry. As the shape update of the FFD block is continuous, there is no relative movement between patches within the FFD block. In the discrete space, we can relate the NURBS control points of the shell patches to the control points of the FFD block,

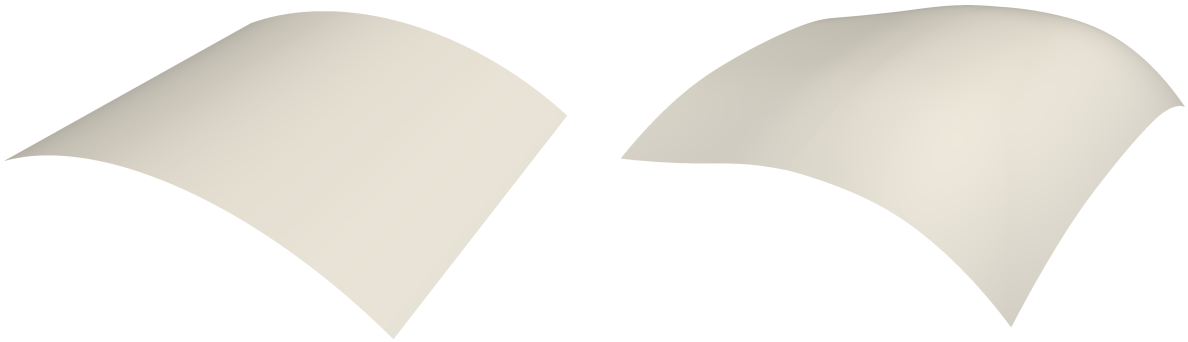
$$\mathbf{N}_{\mathrm{FFD}}\left(\mathbf{S}^{\mathrm{I}}(\boldsymbol{\xi}^{\mathrm{I,FE}})\right)\mathbf{P}_{\mathrm{FFD}} = \mathbf{N}_{\mathrm{FFD}}(\mathbf{P}^{\mathrm{I,FE}})\mathbf{P}_{\mathrm{FFD}} = \mathbf{M}^{\mathrm{I}}\mathbf{P}^{\mathrm{I}} \,. \tag{4.13}$$

The control points of the NURBS surface $\mathbf{P}^{\mathrm{I}}$ of shell patches can be updated through the control points of the FFD block $\mathbf{P}_{\mathrm{FFD}}$. Let $\mathbf{N}_{\mathrm{FFD}}(\mathring{\mathbf{P}}^{\mathrm{I,FE}}) := \mathbf{A}^{\mathrm{I}}_{\mathrm{FFD}}$, where $\mathring{\mathbf{P}}^{\mathrm{I,FE}}$ denotes Lagrange control

points of spline patch I in the baseline configuration. Then $\mathbf{P}^{\mathrm{I}}$ can be computed as

$$\mathbf{P}^{\mathrm{I}} = \left(\left((\mathbf{M}^{\mathrm{I}})^{\mathrm{T}} \mathbf{M}^{\mathrm{I}} \right)^{-1} (\mathbf{M}^{\mathrm{I}})^{\mathrm{T}} \mathbf{A}^{\mathrm{I}}_{\mathrm{FFD}} \right) \mathbf{P}_{\mathrm{FFD}} \, . \tag{4.14}$$

It is noted that we need to solve the system using Moore–Penrose pseudo inverse due to the non-square nature of the extraction matrix $\mathbf{M}^{\mathrm{I}}$, which has dimensions of $n^{\mathrm{I,FE}} \times n^{\mathrm{I}}$. For the extraction matrix, we have $n^{\mathrm{I,FE}} > n^{\mathrm{I}}$, which means that we are solving an overdetermined system. Therefore, $\mathbf{P}^{\mathrm{I}}$ is considered as a least square fit in (4.14) rather than an exact solution. The shape update strategy using FFD block is illustrated in the third step in Figure 4.1, and the resulting shell patches with control net are depicted in the fourth step. A comparison between the initial non-matching cylindrical roof and updated NURBS surfaces is shown in Figure 4.2, where the surface–surface intersections keep overlapping within tolerance in the updated configuration.



(a) **(b)**

Figure 4.2. (a) Initial configuration of the cylindrical roof geometry consisting of four non-matching NURBS patches. (b) Updated NURBS surfaces using FFD block.

The procedures to update control points of non-matching shells with m patches are summarized as follows:

1. In the preprocessing step, generate sparse matrices of evaluation of FFD block B-spline basis functions at shells' Lagrange control points in the initial configuration $\left\{ \mathbf{A}^{\mathrm{I}}_{\mathrm{FFD}} \right\}$ and Lagrange extraction matrices $\left\{ \mathbf{M}^{\mathrm{I}} \right\}$, for $\mathrm{I} \in \{\mathrm{A}, \mathrm{B}, \ldots, \mathrm{I}^{m}\}$.

2. At optimization iteration step i^{opt}, obtain updated control points of the FFD block $\left(\mathbf{P}_{\text{FFD}}\right)^{i^{\text{opt}}}$. Compute updated Lagrange control points $\left(\mathbf{P}^{\text{I,FE}}\right)^{i^{\text{opt}}}$ for all shell patches,

$$\mathbf{A}^{\text{I}}_{\text{FFD}} \left(\mathbf{P}_{\text{FFD}}\right)^{i^{\text{opt}}} = \left(\mathbf{P}^{\text{I,FE}}\right)^{i^{\text{opt}}} . \tag{4.15}$$

3. Solve NURBS control points $\left(\mathbf{P}^{\text{I}}\right)^{i^{\text{opt}}}$ at step i^{opt} through Moore–Penrose pseudo inverse,

$$\mathbf{M}^{\text{I}} \left(\mathbf{P}^{\text{I}}\right)^{i^{\text{opt}}} = \left(\mathbf{P}^{\text{I,FE}}\right)^{i^{\text{opt}}} . \tag{4.16}$$

4. Perform IGA with updated shell geometry, evaluate objective function and derivatives if needed, then proceed with optimization iteration.

Though the control points of the shell patches are computed in the least square fit sense, the updated geometry can still retain the intersection with sufficient discretization. A sliced view of the intersection, in accompaniment with NURBS and Lagrange control points, between the right two spline patches S^{C} and S^{D} in the first step of Figure 4.1 is shown in Figure 4.3, where we use coarser discretizations to make the comparison clearer. In the updated configuration, the two cubic intersecting edges are still overlapping even with only 5 and 6 NURBS control points.

Since there is no relative movement between intersecting spline patches within the FFD block, which can be achieved with adequate control points in the discrete context, parametric coordinates of surface–surface intersections remain unchanged during shape updates. Therefore, transfer matrices introduced in Section 3.3 can be reused to interpolate data from spline patches to quadrature meshes when integrating penalty energies in the optimization iteration. These matrices only need to be generated once at the preprocessing stage.

4.2.1 Sensitivities for shape optimization

By utilizing the capabilities of direct analysis for non-matching isogeometric shells and incorporating FFD-based shape updates, we are able to conduct shape optimization for the shell structures in a seamless manner. The problem that optimizes the shape of non-matching shells

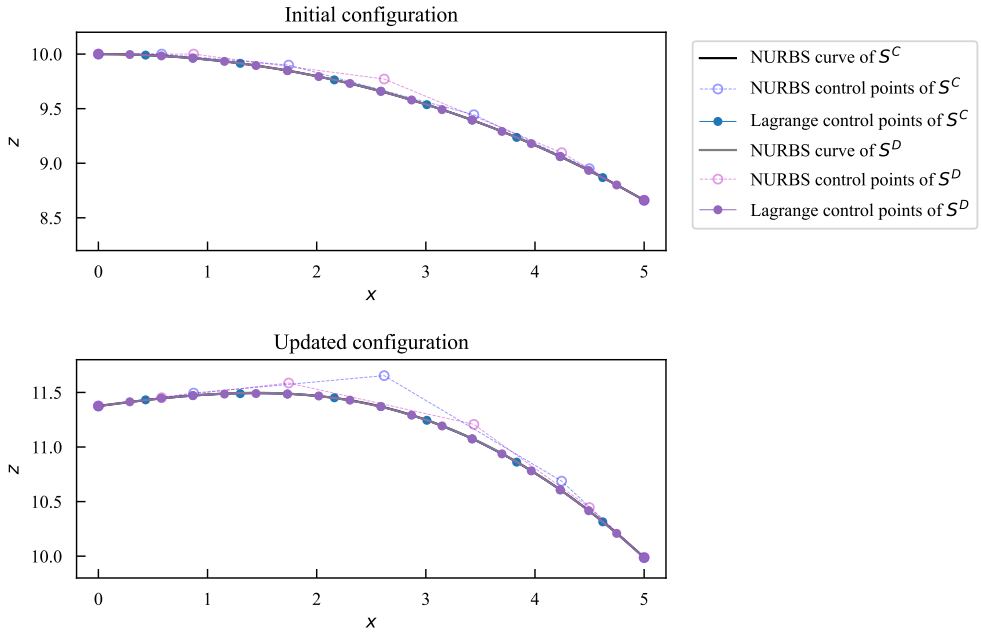


Figure 4.3. Sliced view of the intersecting edges between shell patches S^{C} and S^{D} of the cylindrical roof. The two edges remain overlapping in the updated configuration.

can be formulated as follows,

$$\begin{aligned} \text{minimize} \quad & f\left(\mathbf{P}_{\mathrm{FFD}}\right) \\ \text{subject to} \quad & g_{i_g}\left(\mathbf{P}_{\mathrm{FFD}}\right) \leq \mathbf{0}\,, \text{for } i_g \in \{1,2,\ldots,n_g\} \\ & h_{i_h}\left(\mathbf{P}_{\mathrm{FFD}}\right) = \mathbf{0}\,, \text{for } i_h \in \{1,2,\ldots,n_h\} \\ & \mathbf{P}_{\mathrm{FFD}l} \leq \mathbf{P}_{\mathrm{FFD}} \leq \mathbf{P}_{\mathrm{FFD}u}\,, \end{aligned} \tag{4.17}$$

where control points of the FFD block $\mathbf{P}_{\mathrm{FFD}}$ are design variables, f is the objective function, g_{i_g} are inequality constraints, and h_{i_h} are equality constraints. $\mathbf{P}_{\mathrm{FFD}l}$ and $\mathbf{P}_{\mathrm{FFD}u}$ are lower and upper limits for the design variables.

To perform gradient-based design optimization, we formulate total derivatives of the objective function with respect to design variables as

$$\mathrm{d}_{\mathbf{P}_{\mathrm{FFD}}} f = \left(\partial_{\mathbf{P}} f + \partial_{\mathbf{d}} f \, \mathrm{d}_{\mathbf{P}} \mathbf{d}\right) \mathrm{d}_{\mathbf{P}_{\mathrm{FFD}}} \mathbf{P}\,, \tag{4.18}$$

where $\mathbf{P}^{\mathrm{T}} = \begin{bmatrix} \mathbf{P}^{\mathrm{A}} & \mathbf{P}^{\mathrm{B}} & \dots & \mathbf{P}^{\mathrm{I}^m} \end{bmatrix}$ is the full vector of NURBS control points, and similarly, $\mathbf{d}^{\mathrm{T}} = \begin{bmatrix} \mathbf{d}^{\mathrm{A}} & \mathbf{d}^{\mathrm{B}} & \dots & \mathbf{d}^{\mathrm{I}^m} \end{bmatrix}$ is the full vector of shell displacements in IGA DoFs.

Partial derivatives $\partial_{\mathbf{P}} f$ and $\partial_{\mathbf{d}} f$ in (4.18) can be computed and depend on the objective function combined with extraction matrices,

$$\partial_{\mathbf{P}} f = \mathbf{M}^{\mathrm{T}} \partial_{\mathbf{P}^{\mathrm{FE}}} f \quad \text{and} \quad \partial_{\mathbf{d}} f = \mathbf{M}^{\mathrm{T}} \partial_{\mathbf{d}^{\mathrm{FE}}} f \,, \tag{4.19}$$

where $\mathbf{P}^{\mathrm{FE}^{\mathrm{T}}} = \begin{bmatrix} \mathbf{P}^{\mathrm{A,FE}} & \mathbf{P}^{\mathrm{B,FE}} & \dots & \mathbf{P}^{\mathrm{I}^m,\mathrm{FE}} \end{bmatrix}$, $\mathbf{d}^{\mathrm{FE}^{\mathrm{T}}} = \begin{bmatrix} \mathbf{d}^{\mathrm{A,FE}} & \mathbf{d}^{\mathrm{B,FE}} & \dots & \mathbf{d}^{\mathrm{I}^m,\mathrm{FE}} \end{bmatrix}$.
$\mathbf{M} = \mathrm{diag}(\mathbf{M}^{\mathrm{A}}, \mathbf{M}^{\mathrm{B}}, \dots, \mathbf{M}^{\mathrm{I}^m})$ is a diagonal block matrix for global extraction. Calculation of partial derivatives in (4.19) is automated using FEniCS. Formulation for total derivative $\mathrm{d}_{\mathbf{P}_{\mathrm{FFD}}}\mathbf{P}$ is introduced in (4.14). As for total derivative $\mathrm{d}_{\mathbf{P}}\mathbf{d}$, we have the implicit relation between $\mathbf{P}$ and $\mathbf{d}$,

$$\mathbf{r} = \mathbf{R}(\mathbf{P}, \mathbf{d}) = \partial_{\mathbf{d}} W(\mathbf{P}, \mathbf{d}) = \mathbf{0} \,, \tag{4.20}$$

where W is the total energy of the non-matching shells defined in (3.12). Once an updated $\mathbf{P}$ is obtained, the shell displacements need to be solved using (3.12) until the residual vector $\mathbf{r}$ reaches a tolerance. Thus, $\mathbf{r}$ is supposed to remain as $\mathbf{0}$ despite the change of $\mathbf{P}$, and we have the following derivative

$$\mathrm{d}_{\mathbf{P}}\mathbf{r} = \partial_{\mathbf{P}}\mathbf{R} + \partial_{\mathbf{d}}\mathbf{R}\,\mathrm{d}_{\mathbf{P}}\mathbf{d} = \mathbf{0} \,, \tag{4.21}$$

and the total derivative $\mathrm{d}_{\mathbf{P}}\mathbf{d}$ in (4.18) is given by

$$\mathrm{d}_{\mathbf{P}}\mathbf{d} = -\left(\partial_{\mathbf{d}}\mathbf{R}\right)^{-1} \partial_{\mathbf{P}}\mathbf{R} \,. \tag{4.22}$$

Partial derivative $\partial_{\mathbf{d}}\mathbf{R}$ is equivalent to $\partial_{\mathbf{d}}(\partial_{\mathbf{d}} W)$ and is the stiffness matrix defined in (3.22).

Analogously, we use a pair of shell patches to illustrate the formulation of partial derivative $\partial_{\mathbf{P}}\mathbf{R}$,

$$\partial_{\mathbf{P}}\mathbf{R} = \begin{bmatrix} \partial_{\mathbf{P}^{\mathrm{A}}}(\partial_{\mathbf{d}^{\mathrm{A}}}W) & \partial_{\mathbf{P}^{\mathrm{B}}}(\partial_{\mathbf{d}^{\mathrm{A}}}W) \\ \partial_{\mathbf{P}^{\mathrm{A}}}(\partial_{\mathbf{d}^{\mathrm{B}}}W) & \partial_{\mathbf{P}^{\mathrm{B}}}(\partial_{\mathbf{d}^{\mathrm{B}}}W) \end{bmatrix} . \tag{4.23}$$

Partial derivatives in (4.23) have identical expressions to (3.22).

Extend partial derivatives in (4.23) to shell structures with an arbitrary number of patches, and substitute $\mathrm{d}_{\mathbf{P}}\mathbf{d}$ in (4.18) with (4.22), we can obtain the total derivative of the shape optimization

$$\mathrm{d}_{\mathbf{P}_{\mathrm{FFD}}} f = \left(\partial_{\mathbf{P}} f - \partial_{\mathbf{d}} f \left(\partial_{\mathbf{d}}\mathbf{R}\right)^{-1} \partial_{\mathbf{P}}\mathbf{R}\right) \mathrm{d}_{\mathbf{P}_{\mathrm{FFD}}}\mathbf{P} . \tag{4.24}$$

4.2.2 Sensitivities for thickness optimization

The idea of FFD-based shape update can be applied to shell thickness optimization, where the shell thickness is treated as an extra field of the NURBS control points. We can use (4.14) to build the relation of the thickness between shell patches and FFD block,

$$\mathbf{t}^{\mathrm{I}} = \left(\left((\mathbf{M}^{\mathrm{I}}_{\mathrm{s}})^{\mathrm{T}}\mathbf{M}^{\mathrm{I}}_{\mathrm{s}}\right)^{-1} (\mathbf{M}^{\mathrm{I}}_{\mathrm{s}})^{\mathrm{T}}\mathbf{A}^{\mathrm{I}}_{\mathrm{FFDs}}\right) \mathbf{t}_{\mathrm{FFD}} , \tag{4.25}$$

where $\mathbf{t}^{\mathrm{I}}$ is the thickness for shell S^{I} in IGA DoFs, and $\mathbf{t}_{\mathrm{FFD}}$ is the corresponding thickness field of the FFD block. Subscript s in $\mathbf{M}^{\mathrm{I}}_{\mathrm{s}}$ and $\mathbf{A}^{\mathrm{I}}_{\mathrm{FFDs}}$ denotes matrices for scalar fields. Note that $\mathbf{t}_{\mathrm{FFD}}$ is not the actual thickness of the B-spline solid geometry, but an extra set of the control points on the FFD block to update the thickness of the non-matching shells. Accordingly, the identical shape update strategy is applicable to thickness update. FFD-based thickness optimization also offers the benefit that shell thickness remains continuous on the surface–surface intersections.

Replacing control points of the FFD block in (4.17) with $\mathbf{t}_{\mathrm{FFD}}$, one can have the problem description of thickness optimization. Since both Kirchhoff–Love shell total work W^{A} and

W^{B}, and penalty energy $W^{\mathrm{AB}}_{\mathrm{pen}}$ involve shell thickness, the total derivative and associated partial derivatives of the thickness optimization problem can be acquired by replacing $\mathbf{P}_{\mathrm{FFD}}$, $\mathbf{P}^{\mathrm{FE}}$, $\mathbf{P}$ with $\mathbf{t}_{\mathrm{FFD}}$, $\mathbf{t}^{\mathrm{FE}}$ and $\mathbf{t}$ in Eqs. (4.22)–(4.24), respectively. The total derivative of FFD-based thickness optimization reads as

$$\mathrm{d}_{\mathbf{t}_{\mathrm{FFD}}} f = \left(\partial_{\mathbf{t}} f - \partial_{\mathbf{d}} f \left(\partial_{\mathbf{d}} \mathbf{R} \right)^{-1} \partial_{\mathbf{t}} \mathbf{R} \right) \mathrm{d}_{\mathbf{t}_{\mathrm{FFD}}} \mathbf{t} \, . \tag{4.26}$$

In some applications, one may choose to have a constant thickness for each shell patch. This can be easily achieved within the current framework by relating the shell thickness in IGA DoFs to one scalar value $t^{\mathrm{I,const}}$ as

$$\mathbf{t}^{\mathrm{I}} = \mathbf{c}^{\mathrm{I}} t^{\mathrm{I,const}} \, , \tag{4.27}$$

where $\mathbf{c}^{\mathrm{I}} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^{\mathrm{T}}$ is a unit column vector contains n^{I} entries. The total derivative for piecewise constant thickness optimization can be obtained by replacing $\mathrm{d}_{\mathbf{t}_{\mathrm{FFD}}} \mathbf{t}$ in (4.26) with the following derivative,

$$\mathrm{d}_{\mathbf{t}^{\mathrm{const}}} \mathbf{t} = \mathrm{diag}(\mathbf{c}^{\mathrm{A}}, \mathbf{c}^{\mathrm{B}}, \dots, \mathbf{c}^{\mathrm{I}^m}) \, . \tag{4.28}$$

The FFD block is not needed in piecewise constant only thickness optimization.

These two approaches can be combined together to achieve a more realistic design, where specific sections of the structure necessitate a continuous thickness distribution while constant thickness is better suited for other patches. In the implementation, shell patches can be separated into various groups. One group comprises the shell patches immersed within an FFD block allowing for a continuous thickness distribution. On the other hand, the shell patches not contained in an FFD block are assumed to have a constant thickness. Moreover, shell patches originating from different FFD blocks would exhibit discontinuous thickness at

their intersections. Therefore, the combined thickness optimization approach provides more flexibility.

4.3 Benchmark Problems

A series of benchmark problems are considered to verify the effectiveness of the optimization method. Sections 4.3.1–4.3.3 illustrate that baseline non-matching shell structures with arbitrary intersections are able to accurately converge to the analytical optimum. Section 4.3.4 studies the capability and flexibility of the framework for thickness optimization.

4.3.1 Arch shape optimization

An arch fixed at two ends and subjected to a constant downward load per unit horizontal length is modeled by a Kirchhoff–Love shell theory. Detailed problem definitions can be found in [107, Section 8]. To test the effectiveness of FFD-based shape optimization for the non-matching shells approach, we model the arch using four NURBS patches with three intersections, where the arch geometry in the baseline configuration is shown in Figure 4.4a. We immerse the arch geometry in a trivariate B-spline block in the initial configuration, as is illustrated in Figure 4.4b. The analytical optimal solution is given by a quadratic parabola, where the ratio between the height of the arch and the horizontal distance of two fixed edges is 0.54779.

We use quadratic B-spline for the FFD block in all three directions, $p_{\text{FFD}} = 2$. The arch shell patches are described by cubic NURBS surfaces, $p_{\text{sh}} = 3$, with 1086 DoFs in total. This benchmark problem minimizes the internal energy of the shell structure, with vertical coordinates of the control points of the FFD block being the design variables. From the control net in Figure 4.4b, it can be observed that there are 54 design variables. Two constraints are applied to this problem. The first constraint ensures that the lines in the FFD control net are parallel to the axial direction of the arch, keeping the arch devoid of tilting or twisting during the optimization process. The second constraint fixes the bottom layer of FFD control points so that the two edges of the arch remain in the initial position. We use the SLSQP optimizer with a tolerance of 10^{-12}

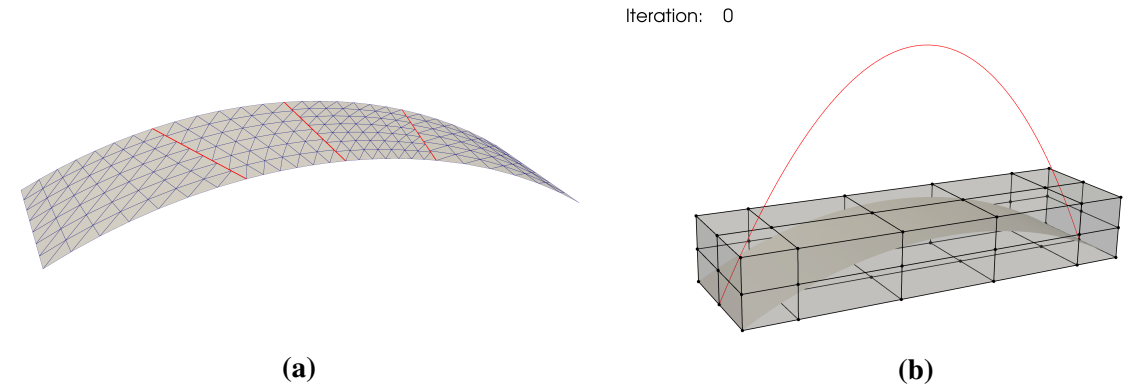


Figure 4.4. (a) Baseline geometry of a non-matching arch consisting of four NRUBS patches, three surface–surface intersections are indicated with red lines. (b) Initial configuration of the arch immersed in an FFD block, where black lines and dots denote the control net. The analytical optimal shape is plotted with a red curve.

to perform the optimization, snapshots of the optimization iteration are demonstrated in Figure 4.5.

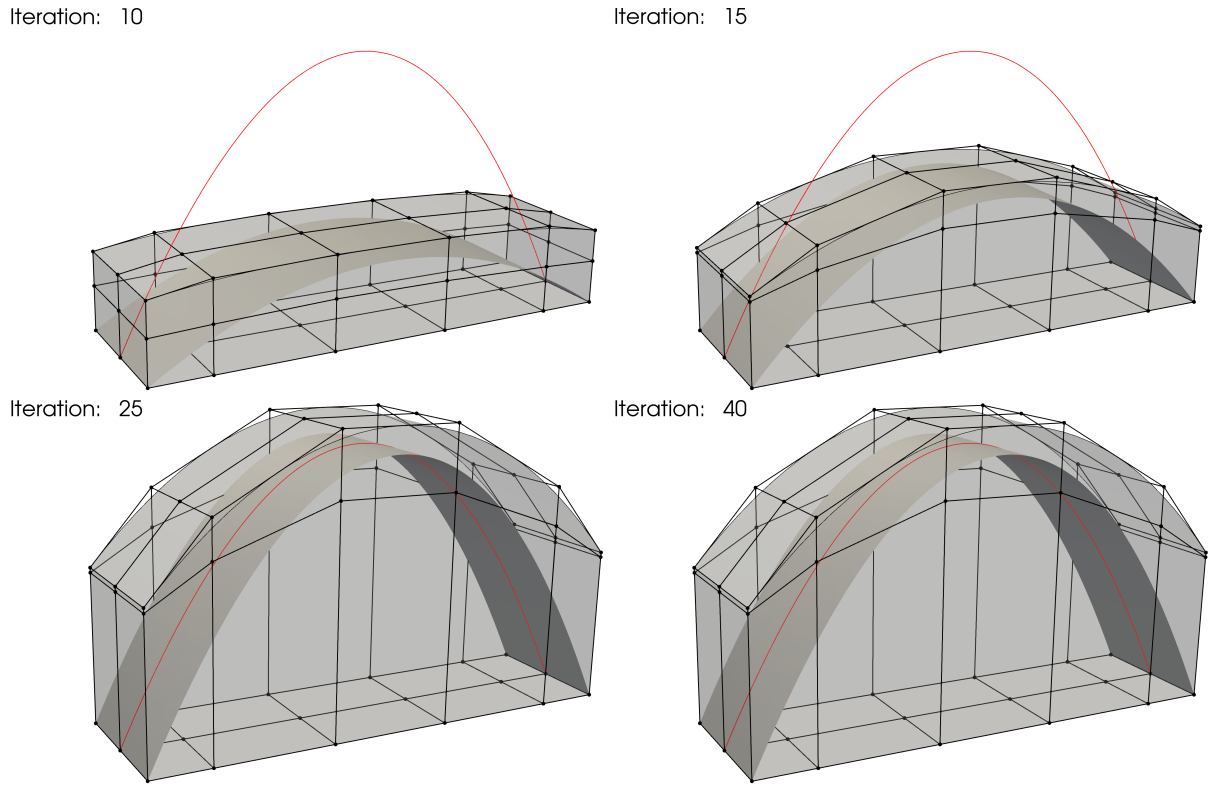


Figure 4.5. Snapshots of non-matching arch shape optimization.

The arch converges to the analytical optimum shape after 40 iterations. The shape of

the FFD block in the final configuration is shown in Figure 4.5. As anticipated, the optimized arch geometry is still contained in the FFD block. The height to base span ratio in this problem is measured as 0.54748, exhibiting a relative error of 0.057% compared to the exact value. Considering the coarse discretization of the arch geometry, the results are encouraging.

4.3.2 Tube shape optimization

A square tube in the baseline configuration is subjected to an internal constant pressure. The optimal shape is given by a cylindrical tube [107, Section 7]. We use four cubic NURBS surfaces to model one-quarter of the square tube, where the initial geometry is illustrated in Figure 4.6a. The square tube geometry contains four non-matching intersections with 1035 DoFs in total, and symmetric boundary conditions are applied in the analysis. The geometry is immersed in a cubic B-spline block to perform FFD-based shape optimization, as shown in Figure 4.6b, where the cross-section of the optimal shape is indicated by the red curve. In this example, the horizontal and vertical coordinates of the FFD control points are design variables, totaling 200 design variables. Similar constraints are employed in this problem as those in the arch shape optimization. Control points on the left and bottom layers are fixed, where the lines in the FFD control net that are parallel to the tube axis maintain their orientations.

(a) **(b)**

Figure 4.6. (a) Baseline geometry of the square tube, a quarter of the tube is modeled using four non-matching B-spline patches with four intersections. (b) Initial configuration of the FFD block with control net. The optimal cross-section is depicted by a red circle.

The optimization problem converges successfully after 42 iterations using SLSQP optimizer with a tolerance of 10^{-12}, and snapshots are depicted in Figure 4.7. As expected, the initial square tube converges to the cylindrical tube.

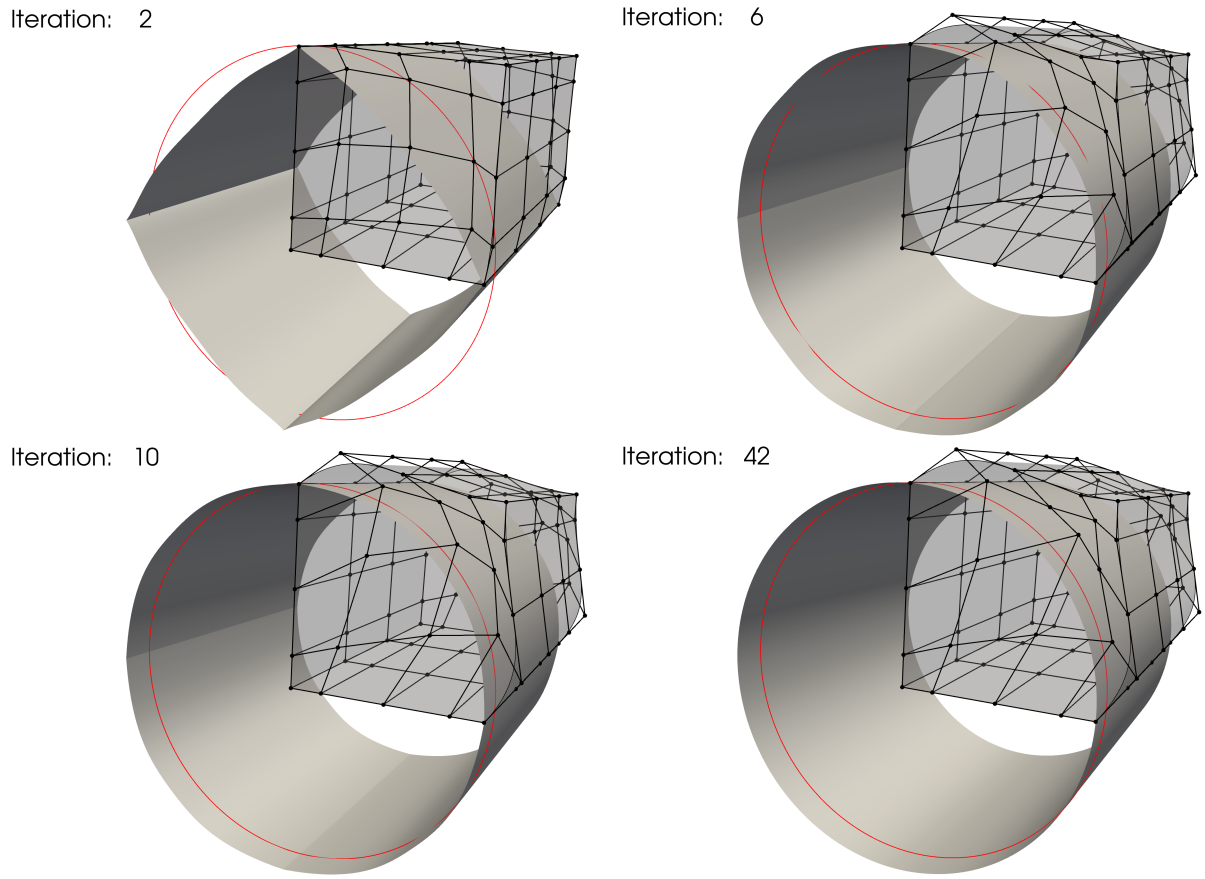
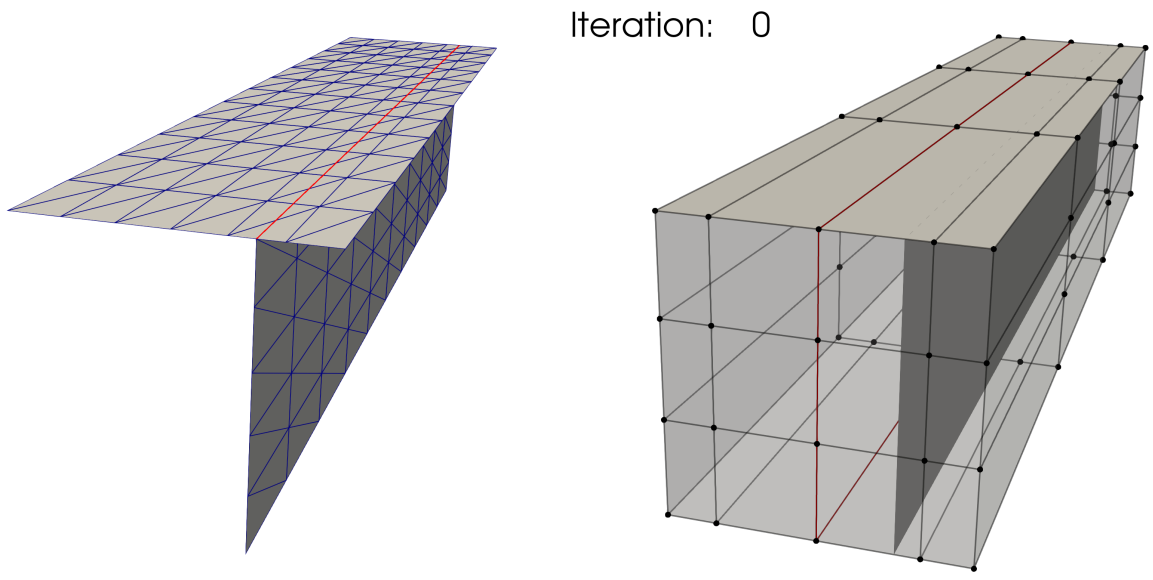


Figure 4.7. Iteration history for tube shape optimization under follower pressure.

4.3.3 T-beam shape optimization

In this section, a T-beam is considered to test the method for shell structures with intersections in the middle. We model a T-beam using two NURBS patches which are shown in Figure 4.8a. In the baseline design, the vertical patch in the T-beam is located at the three-quarter position, where the mismatched intersection is indicated with a red line.

In this benchmark problem, we aim to minimize the internal energy of the T-beam by updating the horizontal coordinates of shell patches' control points. Subsequently, the T-beam is



(a) **(b)**

Figure 4.8. (a) Baseline configuration of a T-beam whose vertical patch is at the three-quarter position of the horizontal patch. (b) The T-beam is placed in an FFD B-spline block. The optimal position of the vertical patch is depicted with red lines.

placed in a cubic FFD block, where the horizontal coordinates of the FFD block's control points serve as design variables. The rear end of the T-beam is fixed. Given a downward distributed load, the optimal shape of the T-beam is expected to have a junction at the center of the horizontal patch under a constant volume constraint. The optimal position of the vertical patch is depicted in Figure 4.8b with red lines. The right and left layers of the FFD block's control points are fixed by employing equality constraints, and control points stay collinear in the axial direction. Using a tolerance of 10^{-15}, the SLSQP optimizer converges successfully after 16 iterations, and the optimization process is shown in Figure 4.9.

In Figure 4.9, the T-beam converges to the optimal solution, indicating that the proposed method is effective for non-matching shell structures with intersections. A more complicated demonstration is presented in Section 6.3, where the geometry contains curved T-junctions.

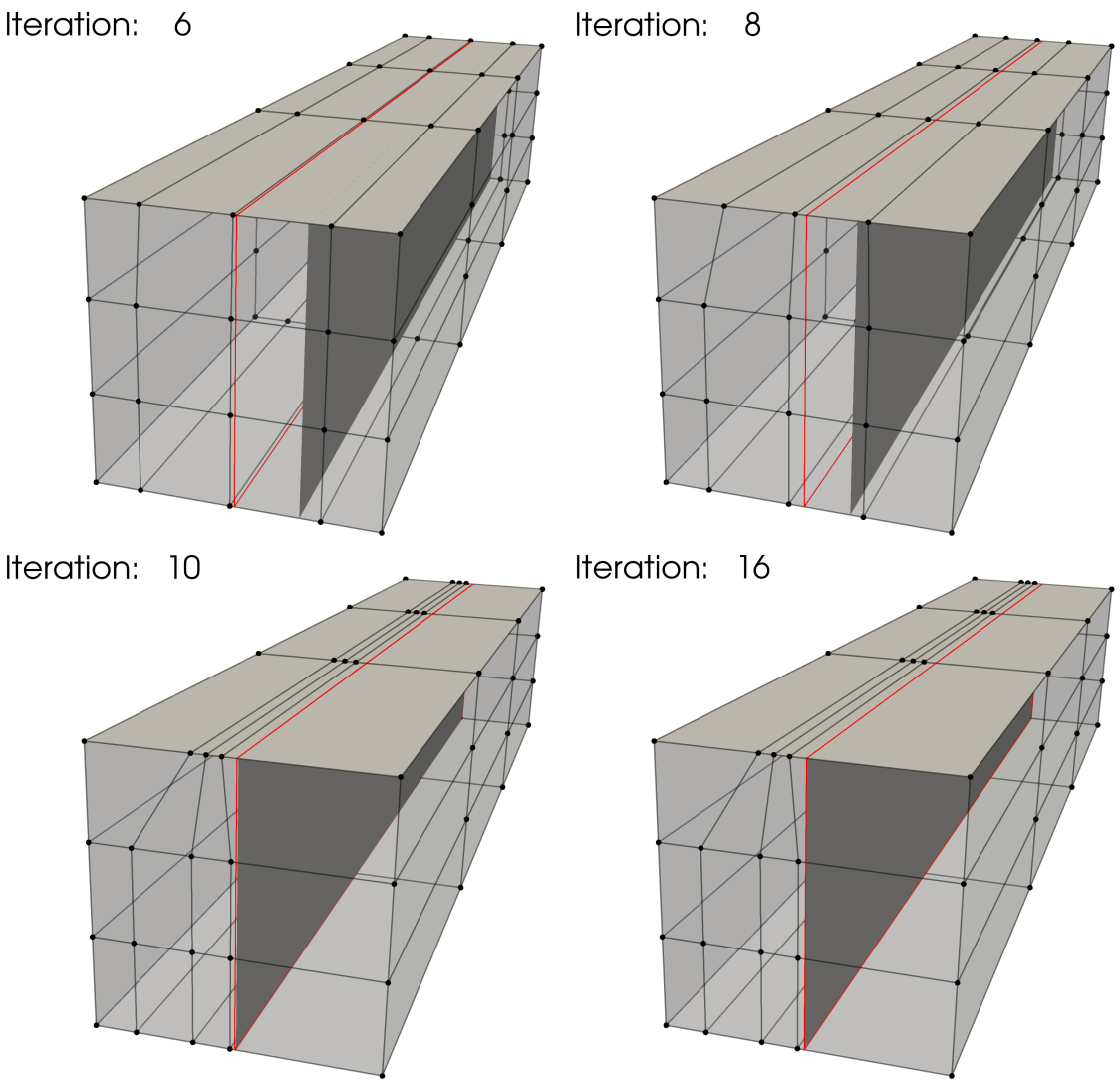


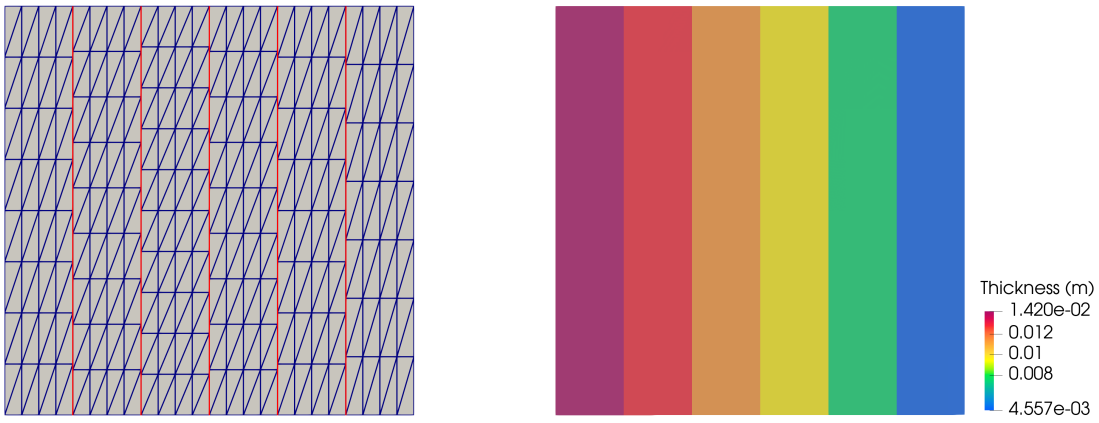
Figure 4.9. Screenshots of T-beam shape optimization process.

4.3.4 Thickness optimization of a clamped plate

As stated in Section 4.2.2, the FFD-based optimization methodology can also be applied to thickness optimization. In the following section, we first demonstrate a piecewise constant thickness optimization for a clamped non-matching plate, in which the FFD block is not needed. Subsequently, we proceed to perform the variable thickness optimization for the same geometry.

Piecewise constant thickness optimization

For the thickness optimization example, a unit square plate composed of six cubic non-matching NURBS surfaces is considered. The geometry, which is shown in Figure 4.10a, exhibits 5 intersections with a total of 1449 DoFs. We apply a clamped boundary condition on the left side and with a line force applied to the right side in the normal direction of the plate. All patches of the plate have an initial thickness of 0.01 m. Using the strategy introduced in Section 4.2.2, we perform piecewise constant thickness optimization for the clamped plate to minimize the internal energy under the constant volume constraint. This problem only has 6 design variables. In this and the following sections, the SNOPT optimizer is used for faster convergence. The optimal thickness is plotted in Figure 4.10b.



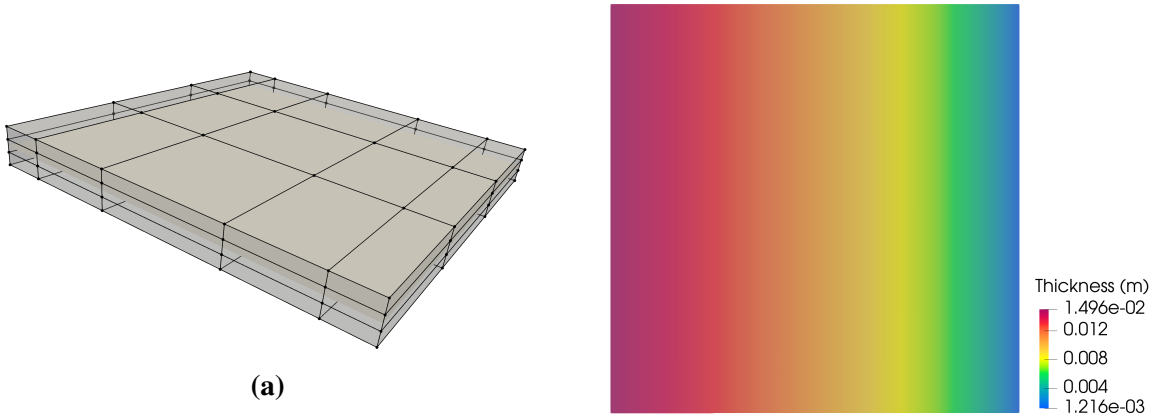
(a) **(b)**

Figure 4.10. (a) A unit square plate consisting of six non-matching patches, intersections are indicated with red lines. (b) Final plate thickness for piecewise constant thickness optimization.

The observed optimal piecewise constant thickness in Figure 4.10b shows material redistributes toward the clamped side, which provides enhanced support to the plate. The internal energy in the final configuration is 37.17% less than the baseline configuration.

Variable thickness optimization

We now perform variable thickness optimization for the non-matching plate. The plate is placed in a cubic B-spline block, which is shown in Figure 4.11a. Besides the constant volume constraint, we only optimize the plate thickness in one direction that is perpendicular to the intersections.

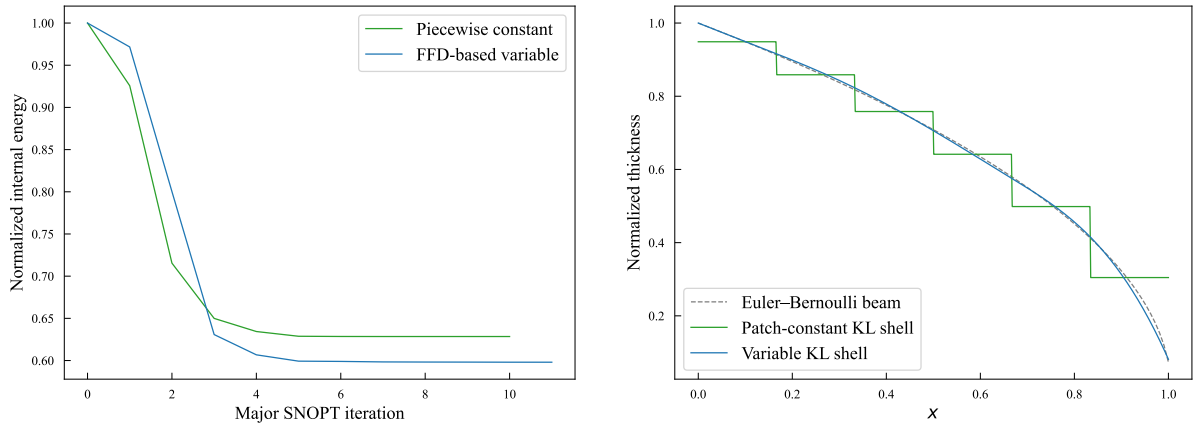


(a) **(b)**

Figure 4.11. (a) The non-matching plate is immersed in an FFD block. (b) Optimized thickness distribution using the FFD-based approach.

With the FFD-based approach, the continuity of shell thickness is maintained at all intersections. Figure 4.11b depicts the converged solution, where a smooth thickness distribution is observed. The smooth thickness profile offers an improved design compared to the piecewise constant thickness approach, resulting in a 40.20% reduction of the initial internal energy. A comparison of optimization iteration of normalized internal energy between the two methods is illustrated in Figure 4.12a. The FFD-based thickness optimization approach converges to a smaller internal energy.

To validate the proposed method, we compare the continuous thickness profile to an optimal thickness configuration of a cantilever beam [163]. The cantilever beam is modeled using the Euler–Bernoulli beam theory, where a point load is applied to the free end. Since the Kirchhoff–Love shell is an extension of the Euler–Bernoulli beam, both models are expected to



(a) **(b)**

Figure 4.12. (a) Optimization process of normalized internal energy for two approaches. (b) Cross-sectional view of piecewise constant thickness and variable thickness, and comparison with Euler-Bernoulli beam thickness optimization.

yield identical thickness distributions. The normalized thickness profiles of these two models, along with the piecewise constant thickness profile, are plotted in Figure 4.12b. A good agreement is observed between the variable thickness of the Kirchhoff–Love shell at the center line and the Euler–Bernoulli beam. Meanwhile, the cross-sectional view of the piecewise constant thickness shows a similar trend to the Euler–Bernoulli beam, albeit with discontinuities at the intersections.

We then investigate the effect of basis function order of continuity in the FFD block. Using the same knots vectors as illustrated in Figure 4.11a, we increase the order of the B-spline basis functions from linear (C^0) to quartic (C^3) and compare the amounts of reduced internal energy relative to the baseline configuration. These data points are summarized in Table 4.1. The results presented in Table 4.1 indicate that an FFD block with quadratic B-spline basis functions can achieve a better optimal thickness distribution for the clamped plate. The internal energy with quadratic FFD block only exhibits 0.27% of relative difference compared to the quartic FFD block. Table 4.1 also suggests that elevating the order of continuity of the FFD block leads to better designs with lower internal energy, particularly when transitioning from linear to quadratic B-spline basis functions.

The plate example demonstrates that both piecewise constant and variable thickness

Table 4.1. Reduction of internal energy of the clamped plate for different degrees of the FFD block.

p_{FFD}	1	2	3	4
Internal energy reduction (%)	39.76	40.11	40.20	40.22

optimization can be conducted in the proposed framework. One can select desired thickness distribution, or a mixed approach of these two, demonstrated in Sections 6.2 and 6.3, based on the physical conditions and problem requirements to obtain an optimal design.

Acknowledgements

A portion of this chapter has been published in "H. Zhao, D. Kamensky, J. T. Hwang, and J. S. Chen. Automated shape and thickness optimization for non-matching isogeometric shells using free-form deformation. *Engineering with Computers*, 1-24, 2024." The dissertation author is the primary investigator and author of this paper.

Chapter 5

Shape optimization with moving intersection

5.1 Shape optimization of non-matching shells with moving intersections

Integrating IGA into shell shape optimization presents notable advantages. The direct analysis based on CAD geometries in IGA naturally bridges the gap between the design model and analysis model within the optimization loop. Compared to the classical FEM, IGA-based shape optimization entirely bypasses the process of conforming FE mesh generation, thereby significantly simplifying the workflow due to the absence of FE mesh sensitivity. This section presents the formulations for IGA-based shape optimization and discrete analytical derivatives [107, 84], followed by an in-depth discussion of multi-patch shell structures with moving intersections.

5.1.1 Shape optimization of isogeometric Kirchhoff–Love shell

A general shape optimization problem for an isogeometric shell patch can be formulated as

$$\begin{aligned} &\underset{\mathbf{P}}{\text{minimize}}\ f(\mathbf{P}) \\ &\text{subject to } \mathbf{g}(\mathbf{P}) \leq \mathbf{0} \\ &\qquad\qquad \mathbf{h}(\mathbf{P}) = \mathbf{0}\,, \end{aligned} \tag{5.1}$$

where design variable $\mathbf{P}$ are the control points of the shell geometry, f is the objective function, $\mathbf{g}$ and $\mathbf{h}$ are the vector-valued inequality and equality constraints, respectively. We adopt internal energy as the objective function to illustrate the optimization scheme. The internal energy of the Kirchhoff–Love shell is a function of both the control points of geometry $\mathbf{P}$ and displacements $\mathbf{d}$, expressed as $f = W_{\mathrm{s}}^{\mathrm{int}}(\mathbf{P}, \mathbf{d}(\mathbf{P}))$. In this study, a gradient-based optimization algorithm is used due to its benefits in efficiency and suitability to large-scale problems. The total derivative of a single patch shell shape optimization is given by the chain rule

$$\mathrm{d}_{\mathbf{P}} f = \partial_{\mathbf{P}} f + (\partial_{\mathbf{d}} f)^{\mathrm{T}} \, \mathrm{d}_{\mathbf{P}} \mathbf{d} \, , \tag{5.2}$$

where the partial derivatives $\partial_{\mathbf{P}} f$ and $\partial_{\mathbf{d}} f$ can be readily calculated with isogeometric discretization in (3.13). The total derivative $\mathrm{d}_{\mathbf{P}} \mathbf{d}$ can be determined by the physical constraint of the Kirchhoff–Love shell theory $\mathbf{R}_{\mathrm{s}}(\mathbf{P}, \mathbf{d}) = \mathbf{0}$ for all input $\mathbf{P}$, which implies

$$\mathrm{d}_{\mathbf{P}} \mathbf{R}_{\mathrm{s}} = \partial_{\mathbf{P}} \mathbf{R}_{\mathrm{s}} + \partial_{\mathbf{d}} \mathbf{R}_{\mathrm{s}} \, \mathrm{d}_{\mathbf{P}} \mathbf{d} = \mathbf{0} \, , \tag{5.3}$$

$$\mathrm{d}_{\mathbf{P}} \mathbf{d} = -(\partial_{\mathbf{d}} \mathbf{R}_{\mathrm{s}})^{-1} \, \partial_{\mathbf{P}} \mathbf{R}_{\mathrm{s}} = -\mathbf{K}_{\mathrm{s}}^{-1} \, \partial_{\mathbf{P}} \mathbf{R}_{\mathrm{s}} \, , \tag{5.4}$$

where $\partial_{\mathbf{P}} \mathbf{R}_{\mathrm{s}}$ represents the partial derivative of the shell residual force vector with respect to geometry control points, and $(\partial_{\mathbf{B}} \mathbf{A})_{ij} = \partial_{\mathbf{B}_j} \mathbf{A}_i$. In the direct method, $\mathrm{d}_{\mathbf{P}} \mathbf{d}$ can be solved with

$$\mathbf{K}_{\mathrm{s}} \, \mathrm{d}_{\mathbf{P}} \mathbf{d} = -\partial_{\mathbf{P}} \mathbf{R}_{\mathrm{s}} \, . \tag{5.5}$$

However, the cost of solving (5.5) scales linearly with the number of design variables. The adjoint method is employed to circumvent the increasing expenses of solving the linear systems in (5.5) with a large number of design variables. Substituting (5.4) into (5.6), the total derivative

states as

$$\mathrm{d}_{\mathbf{P}} f = \partial_{\mathbf{P}} f - (\partial_{\mathbf{d}} f)^{\mathrm{T}} \mathbf{K}_{\mathrm{s}}^{-1} \, \partial_{\mathbf{P}} \mathbf{R}_{\mathrm{s}} = \partial_{\mathbf{P}} f + (\mathrm{d}_{\mathbf{R}_{\mathrm{s}}} f)^{\mathrm{T}} \, \partial_{\mathbf{P}} \mathbf{R}_{\mathrm{s}} \, , \tag{5.6}$$

where $\mathrm{d}_{\mathbf{R}_{\mathrm{s}}} f$ can be solved with the following equation

$$\mathbf{K}_{\mathrm{s}}^{\mathrm{T}} \, \mathrm{d}_{\mathbf{R}_{\mathrm{s}}} f = -\partial_{\mathbf{d}} f \, . \tag{5.7}$$

The number of linear solves in (5.7) equals the number of model outputs and remains independent of the number of design variables. In practical shape optimization scenarios, the number of design variables typically far exceeds the number of outputs. Therefore, the adjoint method is more advantageous for addressing large-scale optimization problems. By solving the total derivative in (5.7) and substituting it into (5.6), the shell geometry with minimum internal energy is obtained when the algorithm satisfies the optimality condition.

5.1.2 Shape optimization of multi-patch isogeometric Kirchhoff–Love shells

Here, we extend the optimization problem (5.1) to encompass multi-patch shell structures, using a two-patch configuration illustrated in Figure 3.2 to demonstrate the optimization approach. For clarity, we continue to use $\mathbf{P}$ and $\mathbf{d}$ to represent the control points for the geometry and displacements of the non-matching shell. Specifically, we define $\mathbf{P}^{\mathrm{T}} = \begin{bmatrix} \mathbf{P}^{\mathrm{A}^{\mathrm{T}}} & \mathbf{P}^{\mathrm{B}^{\mathrm{T}}} \end{bmatrix}$ and $\mathbf{d}^{\mathrm{T}} = \begin{bmatrix} \mathbf{d}^{\mathrm{A}^{\mathrm{T}}} & \mathbf{d}^{\mathrm{B}^{\mathrm{T}}} \end{bmatrix}$. In addition to the change in geometry control points, multi-patch shell structures involve the relative movement between shell patches during shape optimization. To account for this movement, we introduce an additional set of state variables denoted as $\tilde{\boldsymbol{\xi}}^{\mathrm{T}} = \begin{bmatrix} \tilde{\boldsymbol{\xi}}^{\mathrm{A}^{\mathrm{T}}} & \tilde{\boldsymbol{\xi}}^{\mathrm{B}^{\mathrm{T}}} \end{bmatrix}$ as shown in Figure 3.2, representing the parametric coordinates of the patch intersections, into the shape optimization process.

Section 3.3 indicates that, besides the boundary and load conditions, the displacement

field of non-matching shell structures depends not only on the shell geometry but also on the parametric location of patch intersections. This dependence is encapsulated by the shell coupling residual vector $\mathbf{R}(\mathbf{P},\tilde{\boldsymbol{\xi}},\mathbf{u}) = \mathbf{0}$ introduced in (3.22). The total derivative of shape optimization for non-matching shells $\mathrm{d}_{\mathbf{P}} f$ remains the same as given in (5.2). However, the total derivative $\mathrm{d}_{\mathbf{P}}\mathbf{d}$ is obtained by taking the total derivative of the non-matching residual $\mathbf{R}$,

$$\mathrm{d}_{\mathbf{P}}\mathbf{R} = \partial_{\mathbf{P}}\mathbf{R} + \partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R}\,\mathrm{d}_{\mathbf{P}}\tilde{\boldsymbol{\xi}} + \partial_{\mathbf{d}}\mathbf{R}\,\mathrm{d}_{\mathbf{P}}\mathbf{d} = \mathbf{0}\,, \tag{5.8}$$

$$\mathrm{d}_{\mathbf{P}}\mathbf{d} = -(\partial_{\mathbf{d}}\mathbf{R})^{-1}(\partial_{\mathbf{P}}\mathbf{R} + \partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R}\,\mathrm{d}_{\mathbf{P}}\tilde{\boldsymbol{\xi}})\,, \tag{5.9}$$

where $\partial_{\mathbf{d}}\mathbf{R}$ is the stiffness matrix of the non-matching shell, $\partial_{\mathbf{d}}\mathbf{R} = \mathbf{K}$. Similar to the single patch shell, the partial derivative $\partial_{\mathbf{P}}\mathbf{R}$ can be derived from the residual vector of the non-matching shell and has an identical form to $\mathbf{K}$,

$$\partial_{\mathbf{P}}\mathbf{R} = \begin{bmatrix} \partial_{\mathbf{P}^{\mathrm{A}}}\mathbf{R}^{\mathrm{A}}_{\mathrm{s}} + \partial_{\mathbf{P}^{\mathrm{A}}}\mathbf{R}^{\mathrm{A}}_{\mathrm{pen}} & \partial_{\mathbf{P}^{\mathrm{B}}}\mathbf{R}^{\mathrm{A}}_{\mathrm{pen}} \\ \partial_{\mathbf{P}^{\mathrm{A}}}\mathbf{R}^{\mathrm{B}}_{\mathrm{pen}} & \partial_{\mathbf{P}^{\mathrm{B}}}\mathbf{R}^{\mathrm{B}}_{\mathrm{s}} + \partial_{\mathbf{P}^{\mathrm{B}}}\mathbf{R}^{\mathrm{B}}_{\mathrm{pen}} \end{bmatrix}, \tag{5.10}$$

where the blocks related to penalty terms, e.g., $\partial_{\mathbf{P}^{\mathrm{B}}}\mathbf{R}^{\mathrm{A}}_{\mathrm{pen}}$, can be derived from (3.23),

$$\partial_{\mathbf{P}^{\mathrm{B}}}\mathbf{R}^{\mathrm{A}}_{\mathrm{pen}} = (\mathbf{N}^{\mathrm{B}}(\tilde{\boldsymbol{\xi}}^{\mathrm{B}}))^{\mathrm{T}}\partial_{\tilde{\mathbf{P}}^{\mathrm{B}}}\mathbf{R}^{\mathrm{A}}_{\mathrm{pen}} + (\mathbf{N}^{\mathrm{B}}_{,\tilde{\boldsymbol{\xi}}}(\tilde{\boldsymbol{\xi}}^{\mathrm{B}}))^{\mathrm{T}}\partial_{\tilde{\mathbf{P}}^{\mathrm{B}}_{\tilde{\boldsymbol{\xi}}}}\mathbf{R}^{\mathrm{A}}_{\mathrm{pen}}\,. \tag{5.11}$$

In contrast to the single patch shell, the non-matching shells require additional derivatives, as indicated in (5.9), for shape optimization. The partial derivative $\partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R}$ in (5.9) is crucial for differentiating the movement of the intersection during the shape update of shells. This derivative can be obtained from (3.23) since only the penalty terms involve the parametric coordinates of

the patch intersection. The derivative is expressed as

$$\partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R} = \begin{bmatrix} \partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\mathbf{R}^{\mathrm{A}}_{\mathrm{pen}} & \partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{B}}}\mathbf{R}^{\mathrm{A}}_{\mathrm{pen}} \\ \partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\mathbf{R}^{\mathrm{B}}_{\mathrm{pen}} & \partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{B}}}\mathbf{R}^{\mathrm{B}}_{\mathrm{pen}} \end{bmatrix}, \tag{5.12}$$

where the detailed derivations of sub-blocks is illustrated in 5.1.4 using the chain rule. Upon examination of (3.23), it is apparent that the residual vector of the penalty energy $\mathbf{R}^{\mathrm{A}}_{\mathrm{pen}}$ involves the evaluation of the NURBS basis functions and their first derivatives at parametric coordinates of the patch intersection. Note that (5.12) necessitates the second-order derivatives for both shell patches, a condition naturally satisfied by the NURBS functions. Hence, the higher-order continuity in NURBS basis functions not only facilitates direct discretization of the Kirchhoff–Love shell model but also provides a straightforward solution for the relative shell movement in shape optimization problems. This ensures that the optimization process can accurately compute the sensitivities of intersection movements in multi-patch shell structures.

5.1.3 Implicit relation between shell control points and intersections' parametric coordinates

Another derivative that needs to be computed in (5.9) is the total derivative of parametric coordinates of intersections with respect to shell control points, $\mathrm{d}_{\mathbf{P}}\tilde{\boldsymbol{\xi}}$. This derivative accounts for the sensitivity of the intersection location $\tilde{\boldsymbol{\xi}}$ with respect to the shape changes in shell patches. To obtain the analytical derivatives, we establish a relation between $\tilde{\boldsymbol{\xi}}$ and $\mathbf{P}$ through a system of implicit equations. These equations are formulated into a residual vector $\mathbf{R}_{\mathscr{L}}(\mathbf{P},\tilde{\boldsymbol{\xi}})$, which reads

$$\mathbf{R}_{\mathscr{L}}(\mathbf{P},\tilde{\boldsymbol{\xi}}) = \begin{bmatrix} \mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}_i)\mathbf{P}^{\mathrm{A}} - \mathbf{N}^{\mathrm{B}}(\tilde{\boldsymbol{\xi}}^{\mathrm{B}}_i)\mathbf{P}^{\mathrm{B}} = \mathbf{0} \\ L^{\mathrm{A}2}_j - {L^{\mathrm{A}}_{j-1}}^2 = 0 \\ \tilde{\xi}^{\mathrm{A}\backslash\mathrm{B}}_k - 1\backslash 0 = 0 \\ \tilde{\xi}^{\mathrm{A}\backslash\mathrm{B}}_l - 1\backslash 0 = 0 \end{bmatrix} \quad \begin{array}{l} \text{for } i \in \{1,2,\ldots,m\} \\ \qquad j \in \{2,3,\ldots,m-1\} \end{array}, \tag{5.13}$$

where L_j^{A} is the element length of the quadrature mesh $\tilde{\Omega}$ in physical space defined using the Euclidean distance between two adjacent geometric control points of the quadrature mesh

$$L_j^{\mathrm{A}} = \|\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}_{j+1}^{\mathrm{A}})\mathbf{P}^{\mathrm{A}} - \mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}_{j}^{\mathrm{A}})\mathbf{P}^{\mathrm{A}}\|_2 \,. \tag{5.14}$$

The first line of (5.13) signifies that the parametric coordinates $\tilde{\boldsymbol{\xi}}^{\mathrm{A}}$ and $\tilde{\boldsymbol{\xi}}^{\mathrm{B}}$ for node i of $\tilde{\Omega}$ coincide in physical space. This condition ensures the recovery of the same physical intersection curve from the parametric space on sides A and B. The second line of (5.13) imposes constraints on the quadrature mesh, requiring equally spaced geometric control points and a uniform physical mesh size. This equation rules out the presence of very small elements in the quadrature mesh. The first two lines of (5.13) consist of $4m-2$ equations, while there are $4m$ unknowns in $\tilde{\boldsymbol{\xi}}$.

For an arbitrary intersection between two shell patches subjected to elastic deformation, two discrete points on the interaction parametric coordinates $\tilde{\boldsymbol{\xi}}^{\mathrm{A}}$ or $\tilde{\boldsymbol{\xi}}^{\mathrm{B}}$ are located at the edges of the shell surfaces as illustrated in Figure 3.2. The last two items in (5.13) impose such two additional constraints on interaction kinematics where the two edge coordinates have values of either 1 or 0, depending on their parametric location and are denoted using $1\backslash 0$. The parametric coordinate indices k and l take values of $1, 2, 2m-1$, or $2m$. These two conditions force the intersection edge points to move along their respective edges during the shape optimization process. Ultimately, the four conditions presented in (5.13) guarantee a unique set of intersection parametric coordinates for a given pair of shell surfaces. Newton's iteration is utilized to solve (5.13) as the analytical partial derivative $\partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R}_{\mathscr{L}}$ can be formulated.

With the differentiable residual vector $\mathbf{R}_{\mathscr{L}}$, we can obtain the total derivative $\mathrm{d}_{\mathbf{P}}\tilde{\boldsymbol{\xi}}$ using the following expression

$$\mathrm{d}_{\mathbf{P}}\mathbf{R}_{\mathscr{L}} = \partial_{\mathbf{P}}\mathbf{R}_{\mathscr{L}} + \partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R}_{\mathscr{L}}\,\mathrm{d}_{\mathbf{P}}\tilde{\boldsymbol{\xi}} = \mathbf{0}\,, \tag{5.15}$$

$$\mathrm{d}_{\mathbf{P}}\tilde{\boldsymbol{\xi}} = -(\partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R}_{\mathscr{L}})^{-1}\partial_{\mathbf{P}}\mathbf{R}_{\mathscr{L}}\,, \tag{5.16}$$

where the partial derivatives $\partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R}_{\mathscr{L}}$ and $\partial_{\mathbf{P}}\mathbf{R}_{\mathscr{L}}$ can be readily obtained from (5.13). The derivation for these two partial derivatives is demonstrated in 5.1.5.

Substituting (5.10), (5.12) and (5.16) into (5.9), the total derivative of displacements with respect to the geometric control points of the non-matching structures can be obtained. Finally, the total derivative of the non-matching shell shape optimization problem can be computed by substituting (5.9) into (5.2) to yield

$$\mathrm{d}_{\mathbf{P}} f = \partial_{\mathbf{P}} f - (\partial_{\mathbf{d}} f)^{\mathrm{T}} \mathbf{K}^{-1} \left[\partial_{\mathbf{P}}\mathbf{R} - \partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R} \left(\partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R}_{\mathscr{L}}\right)^{-1} \partial_{\mathbf{P}}\mathbf{R}_{\mathscr{L}} \right] . \tag{5.17}$$

The $(\partial_{\mathbf{d}} f)^{\mathrm{T}}\mathbf{K}^{-1}$ term can be effectively computed using the adjoint method discussed in Section 5.1.1. Depending on the shell discretization and number of points on the intersection quadrature mesh, both the direct method and adjoint method can be considered for calculating $\partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R}\left(\partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R}_{\mathscr{L}}\right)^{-1}\partial_{\mathbf{P}}\mathbf{R}_{\mathscr{L}}$.

By computing the total derivative $\mathrm{d}_{\mathbf{P}} f$ in (5.17), the multi-patch shell structural geometry can be updated using optimization algorithms. A schematic demonstration of the shape update during optimization iterations is depicted in Figure 5.1.

5.1.4 Partial derivatives of the non-matching residual

The block matrices in (5.12) are obtained through the application of the chain rule and taking derivatives for the NURBS basis functions. The formulation of the first diagonal block is detailed as follows,

$$\begin{aligned} \partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\mathbf{R}^{\mathrm{A}}_{\mathrm{pen}} = {} & \left(\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}})\right)^{\mathrm{T}(2,3,1)} \tilde{\mathbf{R}}^{\mathrm{A}}_{\mathrm{pen}} + \left(\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}})\right)^{\mathrm{T}} \partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{R}}^{\mathrm{A}}_{\mathrm{pen}} \\ & + \left(\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\mathbf{N}^{\mathrm{A}},_{\boldsymbol{\xi}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}})\right)^{\mathrm{T}(2,3,1)} \tilde{\mathbf{R}}^{\mathrm{A}\boldsymbol{\xi}}_{\mathrm{pen}} + \left(\mathbf{N}^{\mathrm{A}},_{\boldsymbol{\xi}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}})\right)^{\mathrm{T}} \partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{R}}^{\mathrm{A}\boldsymbol{\xi}}_{\mathrm{pen}} , \end{aligned} \tag{5.18}$$

where $\tilde{\mathbf{R}}^{\mathrm{A}}_{\mathrm{pen}} = \partial_{\tilde{\mathbf{d}}^{\mathrm{A}}} W^{\mathrm{AB}}_{\mathrm{pen}}$ and $\tilde{\mathbf{R}}^{\mathrm{A}\boldsymbol{\xi}}_{\mathrm{pen}} = \partial_{\tilde{\mathbf{d}}^{\mathrm{A}}_{\boldsymbol{\xi}}} W^{\mathrm{AB}}_{\mathrm{pen}}$ are residual vectors of the penalty energy on the quadrature mesh $\tilde{\Omega}$ at location $\tilde{\boldsymbol{\xi}}^{\mathrm{A}}$. $\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}})$ and $\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\mathbf{N}^{\mathrm{A}},_{\boldsymbol{\xi}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}})$ are 3D arrays involve the first

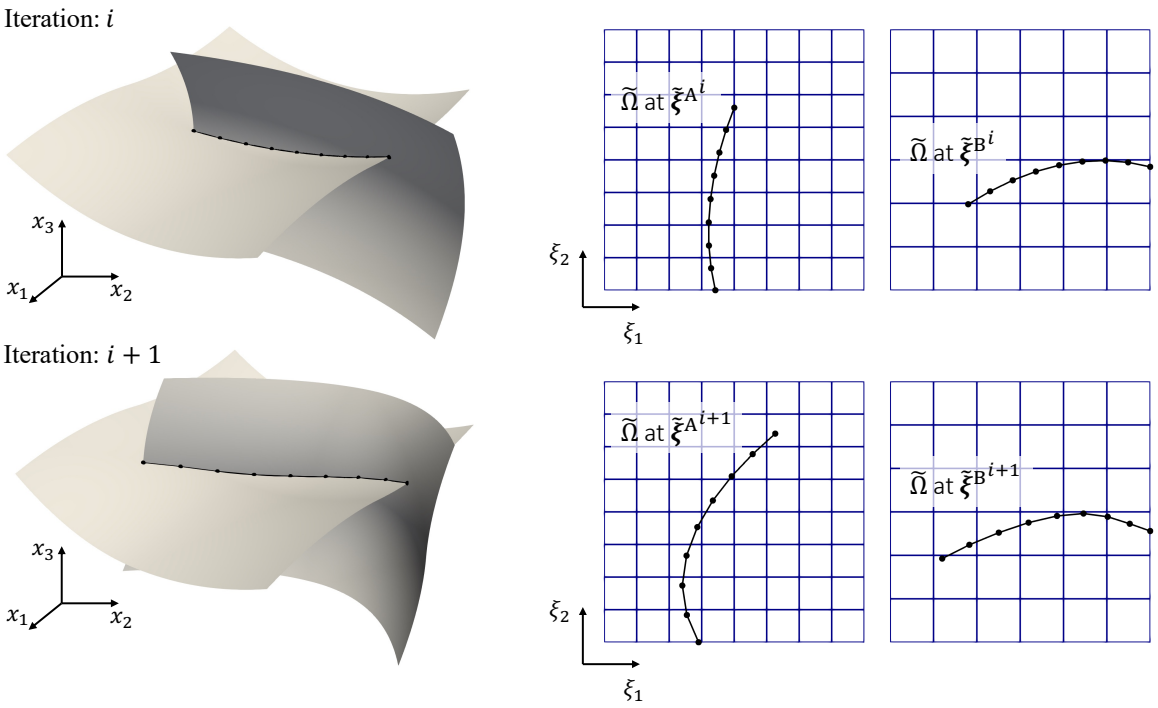


Figure 5.1. Illustration of shape updates and changes in the relative location for two shell patches during shape optimization. The parametric coordinates of the patch intersection are updated from iteration i to $i+1$ accordingly.

order and second order derivatives of the NURBS basis functions, where the transpose $\mathrm{T}(2,3,1)$ indicate switching the axes of the 3D array from $(1,2,3)$ to $(2,3,1)$. Partial derivative $\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{R}}^{\mathrm{A}}_{\mathrm{pen}}$ in (5.18) is formulated as

$$\begin{aligned}\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{R}}^{\mathrm{A}}_{\mathrm{pen}} = \partial_{\tilde{\mathbf{d}}^{\mathrm{A}}}\tilde{\mathbf{R}}^{\mathrm{A}}_{\mathrm{pen}}\,\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{d}}^{\mathrm{A}} + \partial_{\tilde{\mathbf{d}}^{\mathrm{A}}_{\xi}}\tilde{\mathbf{R}}^{\mathrm{A}}_{\mathrm{pen}}\,\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{d}}^{\mathrm{A}}_{\xi} \\ + \partial_{\tilde{\mathbf{P}}^{\mathrm{A}}}\tilde{\mathbf{R}}^{\mathrm{A}}_{\mathrm{pen}}\,\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{P}}^{\mathrm{A}} + \partial_{\tilde{\mathbf{P}}^{\mathrm{A}}_{\xi}}\tilde{\mathbf{R}}^{\mathrm{A}}_{\mathrm{pen}}\,\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{P}}^{\mathrm{A}}_{\xi}\,,\end{aligned} \tag{5.19}$$

where the first component in each term, e.g., $\partial_{\tilde{\mathbf{d}}^{\mathrm{A}}}\tilde{\mathbf{R}}^{\mathrm{A}}_{\mathrm{pen}}$, can be derived from the penalty energy (3.9). For the second components, such as $\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{d}}^{\mathrm{A}}$, it is computed via the interpolation matrix,

$$\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{d}}^{\mathrm{A}} = \partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}(\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}})\,\mathbf{d}^{\mathrm{A}}) = (\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}))^{\mathrm{T}(1,3,2)}\,\mathbf{d}^{\mathrm{A}}\,. \tag{5.20}$$

Likewise, the other component, $\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{d}}^{\mathrm{A}}_{\boldsymbol{\xi}}$, involves the first derivative of the shell displacement is calculated as

$$\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{d}}^{\mathrm{A}}_{\boldsymbol{\xi}} = (\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\mathbf{N}^{\mathrm{A}},_{\boldsymbol{\xi}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}))^{\mathrm{T}(1,3,2)}\,\mathbf{d}^{\mathrm{A}}\,. \tag{5.21}$$

Replacing $\mathbf{d}^{\mathrm{A}}$ with $\mathbf{P}^{\mathrm{A}}$ in (5.20) and (5.21), we can obtain $\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{P}}^{\mathrm{A}}$ and $\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{P}}^{\mathrm{A}}_{\boldsymbol{\xi}}$, and therefore, $\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{R}}^{\mathrm{A}}_{\mathrm{pen}}$. The same derivation can be applied to obtain $\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{R}}^{\mathrm{A}\boldsymbol{\xi}}_{\mathrm{pen}}$. Substituting $\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\mathbf{R}^{\mathrm{A}}_{\mathrm{pen}}$ and $\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{A}}}\tilde{\mathbf{R}}^{\mathrm{A}\boldsymbol{\xi}}_{\mathrm{pen}}$ into (5.18), we can get the first diagonal block matrix of the partial derivative $\partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R}$.

The off-diagonal block has a similar formulation to the diagonal block,

$$\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{B}}}\mathbf{R}^{\mathrm{A}}_{\mathrm{pen}} = (\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}))^{\mathrm{T}}\,\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{B}}}\tilde{\mathbf{R}}^{\mathrm{A}}_{\mathrm{pen}} + (\mathbf{N}^{\mathrm{A}},_{\boldsymbol{\xi}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}))^{\mathrm{T}}\,\partial_{\tilde{\boldsymbol{\xi}}^{\mathrm{B}}}\tilde{\mathbf{R}}^{\mathrm{A}\boldsymbol{\xi}}_{\mathrm{pen}}\,. \tag{5.22}$$

With the diagonal and off-diagonal blocks, we can obtain the partial derivative in (5.12).

In (5.18) and (5.21), it becomes evident that the second-order derivative of the NURBS basis functions for the shell patch is included. Consequently, the C^1 continuity is required if the evaluation point is located at the element boundary. This requirement is inherently fulfilled by the NURBS basis functions, thereby highlighting the advantages of IGA in the application of design optimization.

5.1.5 Partial derivatives of the implicit intersection representation

To compute the analytical total derivative of the non-matching shell optimization problem, partial derivatives of the implicit intersection representation (5.13) with respect to $\mathbf{P}$ and $\tilde{\boldsymbol{\xi}}$ are required. The partial derivative of the implicit residual vector with respect to intersections' parametric coordinates, $\partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R}_{\mathscr{L}}$, states as

$$\partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R}_{\mathscr{L}} = \begin{bmatrix} \mathbf{E}^{\mathrm{A}} & \mathbf{E}^{\mathrm{B}} \\ \mathbf{F}^{\mathrm{A}} & \mathbf{0} \\ & \mathbf{v}_r \\ & \mathbf{v}_s \end{bmatrix}, \tag{5.23}$$

where

$$\mathbf{E}^{\mathrm{A}}_{ik} = \begin{cases} \begin{bmatrix} \mathbf{N}^{\mathrm{A}},_{\xi_1}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}_i)\mathbf{P}^{\mathrm{A}} & \mathbf{N}^{\mathrm{A}},_{\xi_2}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}_i)\mathbf{P}^{\mathrm{A}} \end{bmatrix}, & \text{if } i = k \\ \mathbf{0}, & \text{otherwise} \end{cases} \quad \text{and}$$

$$\mathbf{E}^{\mathrm{B}}_{ik} = \begin{cases} -\begin{bmatrix} \mathbf{N}^{\mathrm{B}},_{\xi_1}(\tilde{\boldsymbol{\xi}}^{\mathrm{B}}_i)\mathbf{P}^{\mathrm{B}} & \mathbf{N}^{\mathrm{B}},_{\xi_2}(\tilde{\boldsymbol{\xi}}^{\mathrm{B}}_i)\mathbf{P}^{\mathrm{B}} \end{bmatrix}, & \text{if } i = k \\ \mathbf{0}, & \text{otherwise} \end{cases}, \tag{5.24}$$

for $i = \{1,2,\ldots,m\}$ and $k = \{1,2,\ldots,m\}$. Matrices $\mathbf{E}^{\mathrm{A}}$ and $\mathbf{E}^{\mathrm{B}}$ have sizes of $(m \times sd) \times (m \times pd)$. Each entry, $\mathbf{E}^{\mathrm{A}}_{ik}$ or $\mathbf{E}^{\mathrm{B}}_{ik}$, is a block matrix with a size of $sd \times pd$. And

$$\mathbf{F}^{\mathrm{A}}_{jk} = \begin{cases} 2\left(\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}_j)\mathbf{P}^{\mathrm{A}} - \mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}_{j-1})\mathbf{P}^{\mathrm{A}}\right)^{\mathrm{T}} \begin{bmatrix} \mathbf{N}^{\mathrm{A}},_{\xi_1}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}_{j-1})\mathbf{P}^{\mathrm{A}} & \mathbf{N}^{\mathrm{A}},_{\xi_2}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}_{j-1})\mathbf{P}^{\mathrm{A}} \end{bmatrix}, & \text{if } k = j-1 \\ -2\left(\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}_{j+1})\mathbf{P}^{\mathrm{A}} - \mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}_{j-1})\mathbf{P}^{\mathrm{A}}\right)^{\mathrm{T}} \begin{bmatrix} \mathbf{N}^{\mathrm{A}},_{\xi_1}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}_{j})\mathbf{P}^{\mathrm{A}} & \mathbf{N}^{\mathrm{A}},_{\xi_2}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}_{j})\mathbf{P}^{\mathrm{A}} \end{bmatrix}, & \text{if } k = j \\ 2\left(\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}_{j+1})\mathbf{P}^{\mathrm{A}} - \mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}_{j})\mathbf{P}^{\mathrm{A}}\right)^{\mathrm{T}} \begin{bmatrix} \mathbf{N}^{\mathrm{A}},_{\xi_1}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}_{j+1})\mathbf{P}^{\mathrm{A}} & \mathbf{N}^{\mathrm{A}},_{\xi_2}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}_{j+1})\mathbf{P}^{\mathrm{A}} \end{bmatrix}, & \text{if } k = j+1 \\ \mathbf{0}, & \text{otherwise} \end{cases}, \tag{5.25}$$

for $j \in \{2,3,\ldots,m-1\}$ and $k \in \{1,2,\ldots,m\}$. $\mathbf{F}^{\mathrm{A}}$ has dimensions of $(m-2) \times (m \times pd)$, with each entry $\mathbf{F}^{\mathrm{A}}_{jk}$ being of size of $1 \times pd$. Additionally, $\mathbf{v}_r$ is a row vector consisting of zero values except for the r-th entry, which is set to 1. And $\mathbf{v}_s$ possesses these same properties.

The partial derivative of the residual vector with respect to shell patches' control points $\partial_{\mathbf{P}}\mathbf{R}_{\mathscr{L}}$ is expressed as

$$\partial_{\mathbf{P}}\mathbf{R}_{\mathscr{L}} = \begin{bmatrix} \mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}^{\mathrm{A}}_i) & -\mathbf{N}^{\mathrm{B}}(\tilde{\boldsymbol{\xi}}^{\mathrm{B}}_i) \\ \mathbf{G}^{\mathrm{A}}_j & \mathbf{0} \\ \mathbf{0} & \\ \mathbf{0} & \end{bmatrix}, \tag{5.26}$$

for $i \in \{1,2,\ldots,m\}$ and $j \in \{2,3,\ldots,m-1\}$. And $\mathbf{G}^{\mathrm{A}}_j$ is a row vector and has the following

definition

$$\mathbf{G}_j^{\mathrm{A}} = 2\mathbf{P}^{\mathrm{A}\,\mathrm{T}} \left[\left(\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}_{j+1}^{\mathrm{A}}) - \mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}_{j}^{\mathrm{A}}) \right)^{\mathrm{T}} \left(\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}_{j+1}^{\mathrm{A}}) - \mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}_{j}^{\mathrm{A}}) \right) - \left(\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}_{j}^{\mathrm{A}}) - \mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}_{j-1}^{\mathrm{A}}) \right)^{\mathrm{T}} \left(\mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}_{j}^{\mathrm{A}}) - \mathbf{N}^{\mathrm{A}}(\tilde{\boldsymbol{\xi}}_{j-1}^{\mathrm{A}}) \right) \right] . \tag{5.27}$$

Consequently, we can obtain the total derivative of the intersections' parametric coordinates with respect to shell patches' control points by substituting (5.23) and (5.26) into (5.16).

5.2 Shape optimization implementation details

The following sections illustrate the implementation details of non-matching shell shape optimization. We adopt the multilevel design concept and present the treatment of various types of intersections. Furthermore, we introduce the dependencies of the open-source Python library used in this paper.

5.2.1 Multilevel design for IGA-based optimization

In this paper, we apply the multilevel design concept [128, 129, 107] to create a flexible design space. The optimizer modifies only the shape of shell structures with coarse discretizations, referred to as the design model, by adjusting the coordinates of their control points. Meanwhile, a refined geometry, named the analysis model, is used for accurate analysis of the structural response after shape modifications. Specifically, for CAD geometries defined using NURBS basis functions, order elevation (p-refinement), knot refinement (h-refinement), and the combination of these two methods (k-refinement) can be employed to produce finer models while preserving the original geometry. This capability in IGA is particularly beneficial for shape optimization problems as it allows the dimension of the design space to be chosen independently from the dimension of the analysis model.

Figure 5.2 presents an example of the multilevel design approach for a single patch shell.

A quadratic surface with coarse discretization defines the design model. Firstly, the order of the B-spline basis functions is increased to cubic through order elevation. Then the surface is refined by inserting a series of knots in both parametric directions to create the fine analysis model. The design model, with significantly fewer degrees of freedom (DoFs) than the analysis model, enhances convergence for optimization problems. Design engineers have the flexibility to define the dimension of the design space by selecting the initial knot vector. Additionally, the continuity of the basis functions in the analysis model can be increased through k-refinement, which is advantageous for problems with higher-order governing equations. Linear operations can achieve k-refinement by creating associated matrices. Commonly used algorithms for implementing these refinement strategies are detailed in the NURBS book [138, Chapter 5].

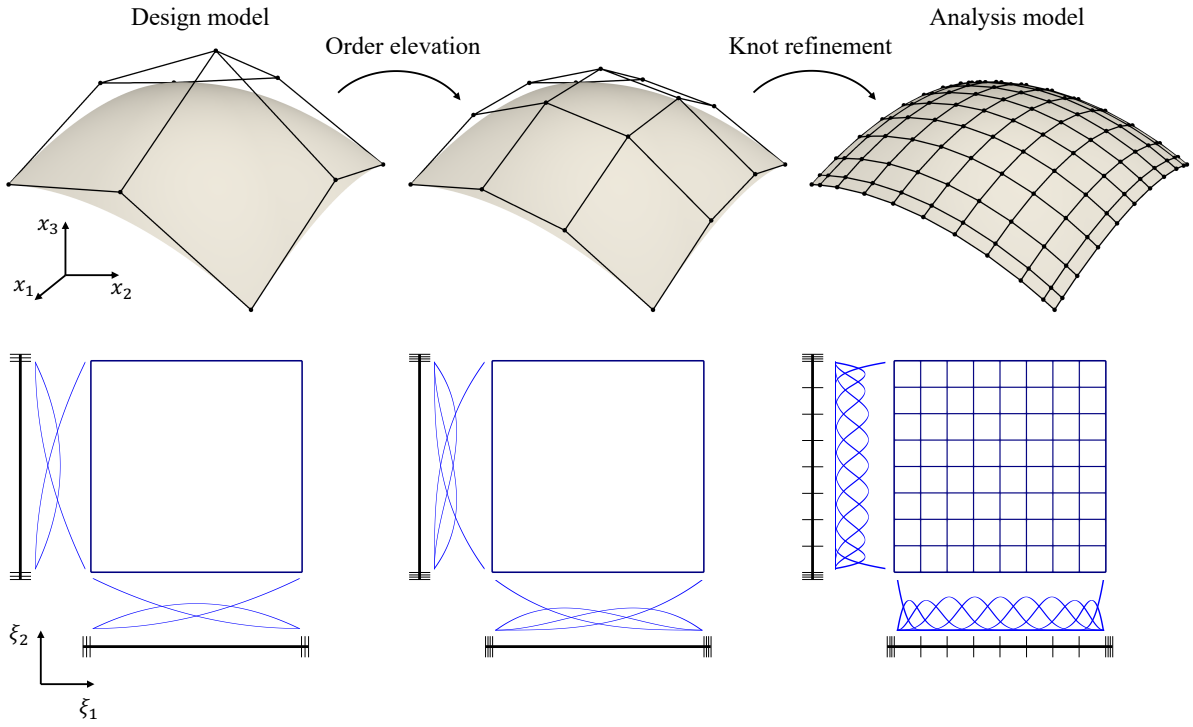


Figure 5.2. Multilevel design approach for shape optimization problems. The coarse design model is employed to update the shape of the geometry, while the refined analysis model is used for structural analysis. Both the design model and the analysis model represent the same geometry.

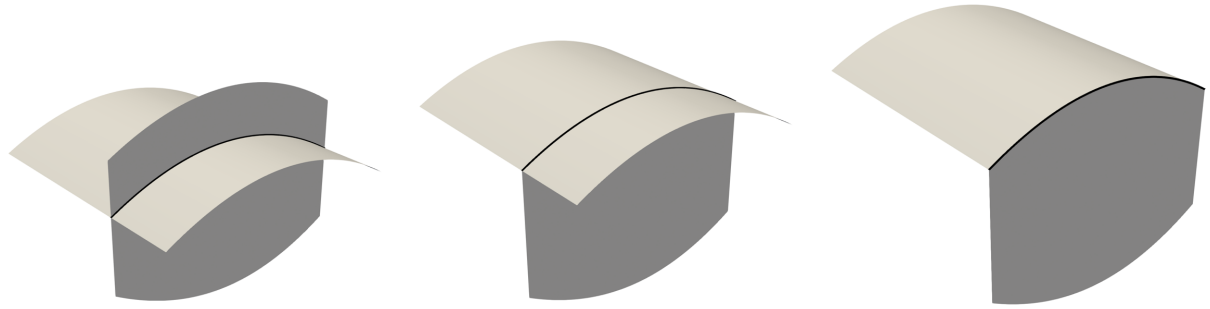
The multilevel design approach can be readily extended to shape optimization with

non-matching shell structures, where the differentiation for the movement of patch intersections during the optimization process is discussed in Section 5.1.2.

5.2.2 Intersection types in shape optimization

Without considering extreme cases such as singular points and singular curves, there are typically three types of intersections between two tensor-product NURBS patches, as shown in Figure 5.3. The first type, named interior–interior intersection, is depicted in Figure 5.3a. For the interior–interior intersections, we assume that two shell patches can move independently of each other without any other constraints imposed. The second type, termed as interior–edge intersection and illustrated in Figure 5.3b, occurs when the edge of one shell patch intersects the interior of the other shell patch, forming a T-junction structure. During the optimization process, the intersection is allowed to move while maintaining the T-junction. Therefore, an additional constraint is necessary to fulfill this requirement. For the third intersection type, as shown in Figure 5.3c, the edges from two separate patches join together and no relative movement between the two patches is allowed. In this intersection topology, the optimization framework enforces the conditions that the relative location of the intersection remains fixed and the two shell patches are always connected at their edges. While these intersection topologies do not represent all possible geometries, they are effective within our targeted applications, particularly in the context of aircraft wing design.

For the interior–edge type of intersections, a linear constraint is applied to the parametric coordinates of the intersection to retain the T-junction. Figure 5.4 depicts the associated parametric configuration and the intersection's quadrature mesh of Figure 5.3b. To preserve the T-junction, the quadrature mesh related to the vertical patch needs to stay on the top edge. Assuming the parametric domain of the vertical patch is a unit square and the lower-left corner is at $(0,0)$, the constraint is applied as $\tilde{\xi}^{\mathrm{B}}_{i2} = 1$ for $i \in \{1,2,\ldots,m\}$. In the example shown in Figure 5.4, the intersecting edge of the vertical patch is only defined by three DoFs, leading to an over-constrained system since $m > 3$. Therefore, we select three points, highlighted in red,



(a) Intersection type: interior–interior **(b)** Intersection type: interior–edge **(c)** Intersection type: edge–edge

Figure 5.3. Types of shell patch intersection in shape optimization problems.

in the quadrature mesh to enforce the T-junction constraint. The support of each NURBS basis function at the intersecting edge needs to contain at least one selected point to uniquely define the edge. It is noted that the edge alignment of the vertical and horizontal patches is imposed only at the selected points to avoid an over-constrained condition. Given the potential for high-order polynomial intersections between two shell patches, the determined curve is considered as an approximated intersecting edge within the design space.

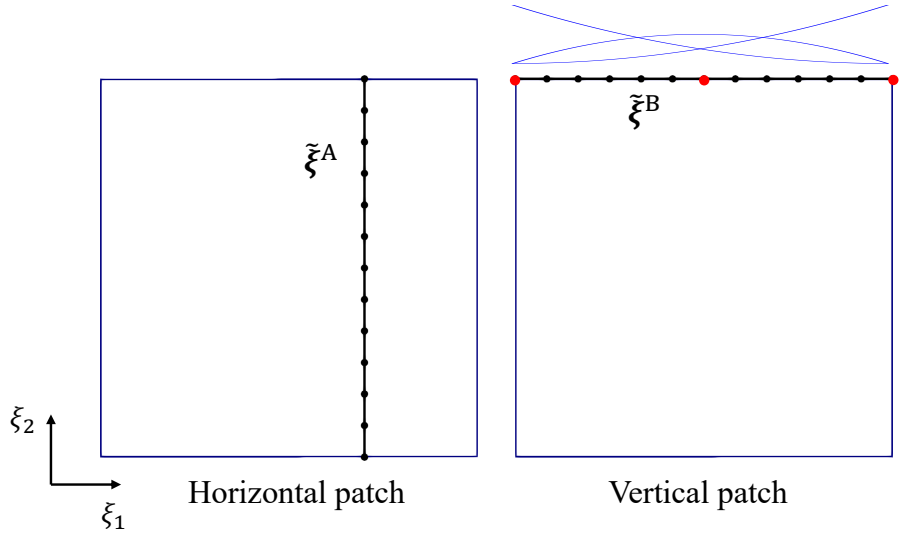


Figure 5.4. Parametric configuration of two shell patches with an interior–edge intersection.

In cases where shell patches form edge-edge intersections, the coordinates of the quadrature mesh are assumed to remain unchanged throughout the optimization process. If the op-

timization problem incorporates the shape of these shell patches, we employ the FFD-based method as proposed in [180, Section 4] to ensure the connectivity between shell patches. The shell patches with edge–edge intersections are embedded within a trivariate B-spline block where the shape of shells is updated through the change of the 3D B-spline block. Meanwhile, parametric coordinates of intersections between shell patches in different B-spline blocks are allowed to move. This strategy is employed in the tube optimization benchmark problem in Section 5.3.2. Conversely, if the shell patches with edge–edge intersections are not considered in the optimization problem, their control points can be fixed without any updates.

5.2.3 Optimization scheme

With the aforementioned implementation details and code dependencies, the workflow of shape optimization for non-matching shells is outlined in Figure 5.5. The optimization workflow entirely bypasses the FE mesh generation for the CAD geometry. Shape modifications are directly applied to the coarse design model, and the structural response of the updated geometry is evaluated using the refined analysis model. As such, the dimension of the design space can be significantly reduced. As discussed in Section 5.2.1, the geometry preservation properties of NURBS surface refinement methods ensure that no geometric errors are introduced from the design model to the analysis model, which is difficult to achieve in traditional FEM. Consequently, this optimization approach guarantees both accurate geometry representation and analysis results.

5.2.4 Software elements for open-source implementation

The shape optimization Python library is developed leveraging a suite of open-source code packages. It employs the Python interface of OpenCASCADE, PythonOCC [137], to import the CAD geometry in IGES or STEP formats into the optimization process. Meanwhile, the surface–surface intersection approximation functionality in PythonOCC is utilized to determine the parametric coordinates of intersections, which serve as the initial guess for (5.13). For

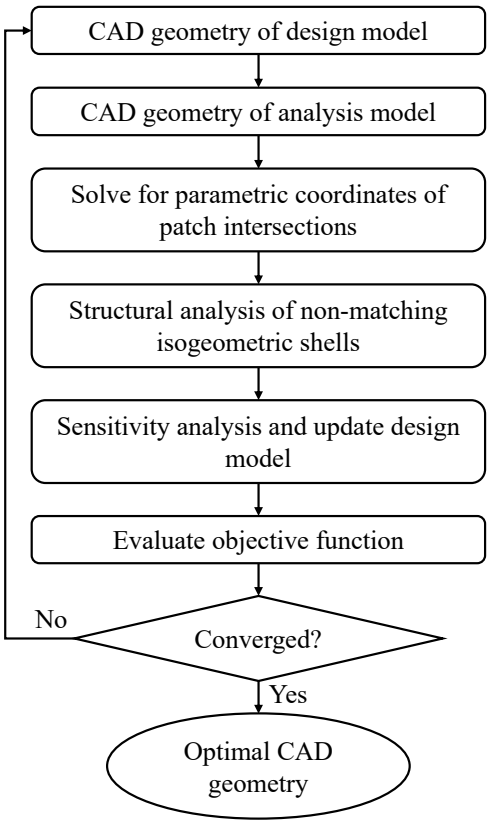


Figure 5.5. Workflow of the IGA-based shape optimization for non-matching shell structures with moving intersections.

automated structural analysis of CAD geometries consisting of non-matching isogeometric Kirchhoff–Love shells, the FEniCS [122]-based library PENGoLINS [181] is employed. The Lagrange polynomial basis functions in the finite element code of FEniCS are changed to NURBS basis functions through the extraction technique [23, 150, 148, 54]. The Lagrange extraction is implemented in tIGAr [98], while the low-level assembly subroutines in FEniCS are reused in the analysis framework.

FEniCS makes use of advanced code generation and computer algebra to automate analytical Gateaux derivative computation, allowing for large-scale gradient-based optimization. Partial derivatives in (5.17) are encapsulated into individual components, and they are modularized through OpenMDAO [63] to manage the adjoint method of total derivative calculation. For solving the optimization problem, the SLSQP optimizer [113] is used for simple benchmark

examples. The SNOPT optimizer [60], renowned for its efficiency in nonlinear problems where gradient evaluations are computationally intensively, is employed for complicated problems. The sparse sequential quadratic programming (SQP) algorithm is used in the SNOPT optimizer. The source code of the shape optimization framework is publicly available on the GitHub repository GOLDFISH [162], where demonstrations presented in Sections 5.3 and 6.4 can be reproduced.

5.3 Benchmark problems

In this section, we present results based on a set of shape optimization problems to validate the effectiveness of the proposed optimization scheme. The multilevel design approach is employed in the T-beam example, while the FFD-based method, which maintains edge–edge intersections, is tested in the tube problem.

5.3.1 T-beam under distributed load

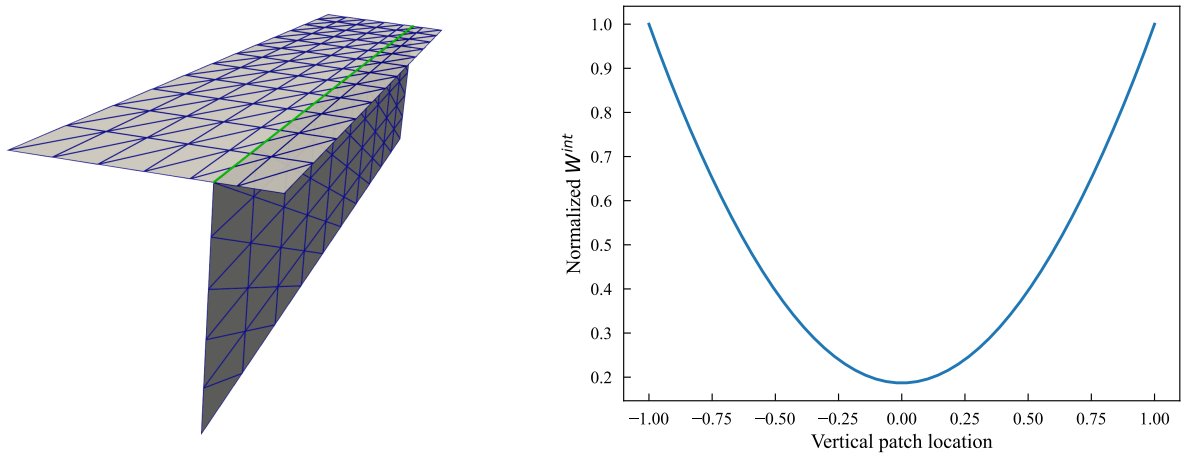
Two types of T-beam geometry are demonstrated to verify the accuracy of the shape optimization approach. The T-beam geometry in Section 5.3.1 has a flat top surface, while the top surface in Section 5.3.1 is curved to test the proposed approach's ability to preserve the T-junction in curved structure in the optimization process. In both demonstrations, the T-beam is subjected to a downward distributed pressure and is fixed at the rear end.

Flat T-beam

For the first benchmark problem, we consider a T-beam geometry composed of two patches, a top surface and a vertical surface. In the optimization process, both surfaces remain flat, with dimensions of 2 m in width and 10 m in length for each patch. The thickness of both shell patches is set as 0.1 m. In the initial design, the top surface ranges from -1 m to 1 m in the horizontal direction, while the top edge of the vertical patch is located at 0.5 m horizontal location. The isogeometrically discretized[1] analysis model using cubic B-spline basis functions

[1]Due to technical limitations within FEniCS, the interpolation matrix described in (3.17) can only be constructed with triangular meshes in the current implementation. While all numerical examples are discretized using triangular

is shown in Figure 5.6a, where the interior–edge intersection is indicated with a green line. Material properties, Young's modulus $E = 10^7$ Pa and Poisson's ratio $\nu = 0$, are used in the analysis, and the uniformly distributed load has a magnitude of $P = 1$ Pa.



(a) **(b)**

Figure 5.6. (a) The isogeometrically discretized T-beam geometry with a flat top surface in the initial configuration. The movable intersection is highlighted with a green line. (b) The T-beam's internal energy depends on the location of the vertical surface. The minimal internal energy occurs when the vertical surface is located at the center of the top surface.

In this benchmark problem, we aim to minimize the internal energy of the T-beam by adjusting the position of the vertical patch. Thus, only one design variable is considered in this problem. The relation between the internal energy of the T-beam and the location of the vertical patch is illustrated in Figure 5.6b. The lowest normalized internal energy, with a value of 0.18719, corresponds to the vertical patch positioned at the center of the top patch. Since the movement of the vertical patch is restricted to the horizontal direction, the requirement for the maintenance of the T-junction is automatically satisfied, and the volume of the T-beam remains constant. The only required constraints in this problem are the limits for the coordinate of the vertical patch, which ranges from -1 m to 1 m. The SLSQP optimizer is adopted for this problem with a tolerance set as 10^{-15}. Due to the simplicity of this benchmark example, the optimizer

meshes, the solutions are still approximated using NURBS basis functions.

converges to the optimal location rapidly and terminates successfully with 4 iterations. The optimized T-beam geometry are demonstrated in Figure 5.7. The vertical patch of the optimized geometry has a horizontal coordinate of 1.323×10^{-9}, closely matching the theoretical optimal solution of 0 with a negligible difference. The normalized internal energy of the converged solution has a value of 0.18721, which shows good agreement with the expected value.

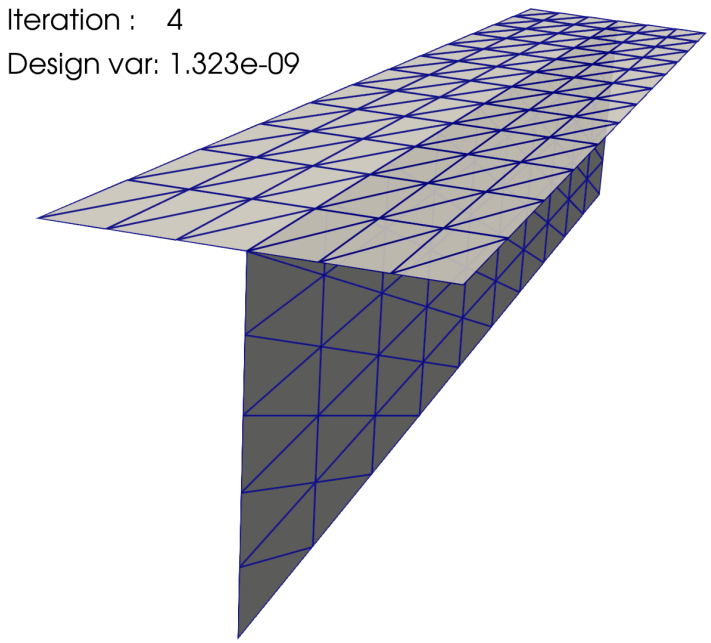


Figure 5.7. Optimized geometry of the T-beam with a flat top surface. The optimizer with a tolerance of 10^{-15} terminates after 4 iterations.

Curved T-beam

For this purpose, a T-beam CAD geometry with a curved top surface is generated, and the associated analysis model discretized with cubic B-spline basis functions is shown in Figure 5.8. The top surface ranges horizontally from -1 m to 1 m and vertically from 0 m to 0.3 m. The vertical surface is located at 0.5 m horizontal location in the initial configuration, where the intersection is marked by a green line. In this benchmark problem, the dimensions of the design space are increased. In the design model, we employ a cubic B-spline surface with a knot vector of $[0,0,0,0,1,1,1,1]$ on both parametric directions to define the horizontal position of

the vertical patch, alongside a linear B-spline curve with a knot vector $[0,0,1,1]$ for its vertical location. This problem involves 16 horizontal design variables and 1 design variable for the vertical location. The same material parameters and objective functions as in Section 5.3.1 are used. A constraint ensuring that the top edge of the vertical surface remains attached to the top surface during the optimization is introduced by fixing the parametric coordinate of the quadrature mesh with respect to the vertical patch to 1.0 in the ξ_2 direction. Additionally, a volume constraint is imposed on the vertical surface to ensure a constant volume.

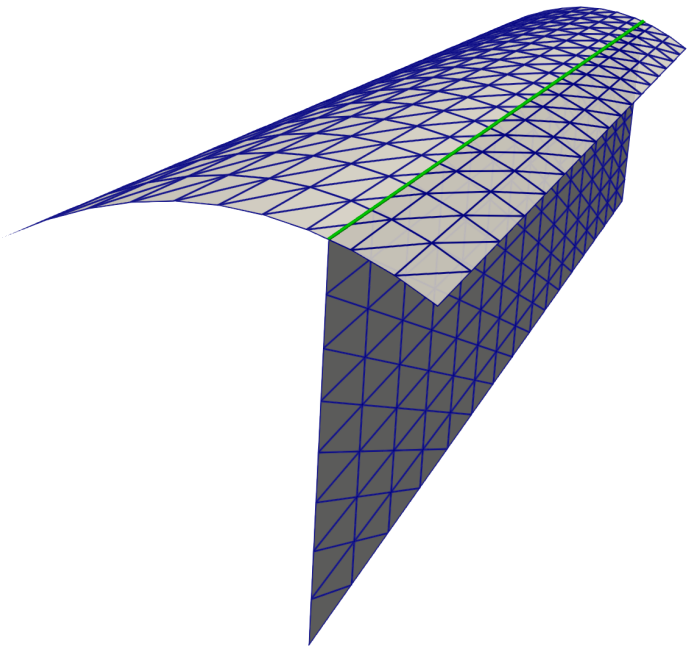


Figure 5.8. Initial configuration of a T-beam geometry with a curved top surface, where the green line indicates the initial location of the intersection.

We employ the SNOPT optimizer with a tolerance of 10^{-6}. It takes 49 iterations for the optimizer to converge to the specified tolerance. A series of representative optimization snapshots of this benchmark problem is shown in Figure 5.9, which demonstrates that the top edge of the vertical surface adheres to the top surface due to the implementation of the T-junction preservation constraint. Despite the increased dimension of the design space allowing for potential bending of the vertical patch, it eventually converges to a flat surface in the optimal configuration to minimize internal energy. The limits of the horizontal control points in the

optimized design are -4.079×10^{-5} and 3.651×10^{-5}, which correspond to the flat vertical surface at the center of the top surface with sufficiently small errors. The flat shape in the optimized design indicates that the vertical surface's volume remains unchanged. Meanwhile, the vertical coordinate of the control point on the top edge in the optimal design of 0.3 confirms that the top edge of the vertical surface precisely lies in the middle of the top surface, maintaining the T-junction connection.

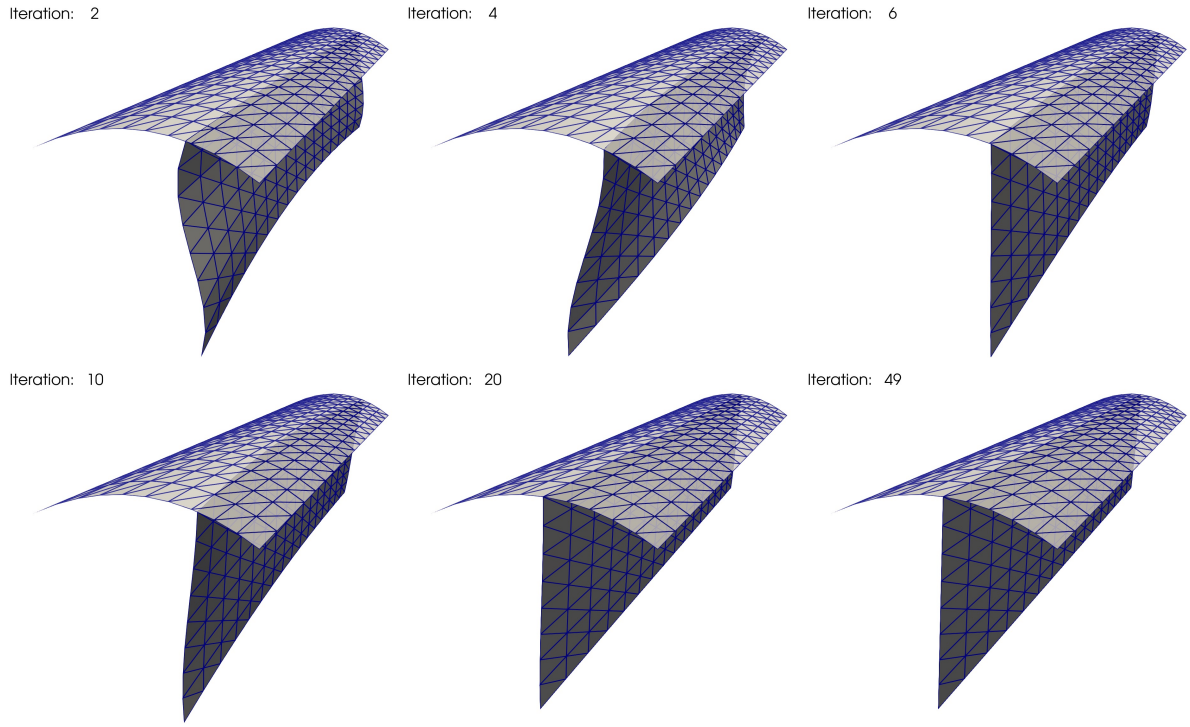


Figure 5.9. Snapshots of the shape optimization history of the T-beam featuring a curved top surface. The SNOPT optimizer requires 49 iterations to converge to the tolerance of 10^{-6}.

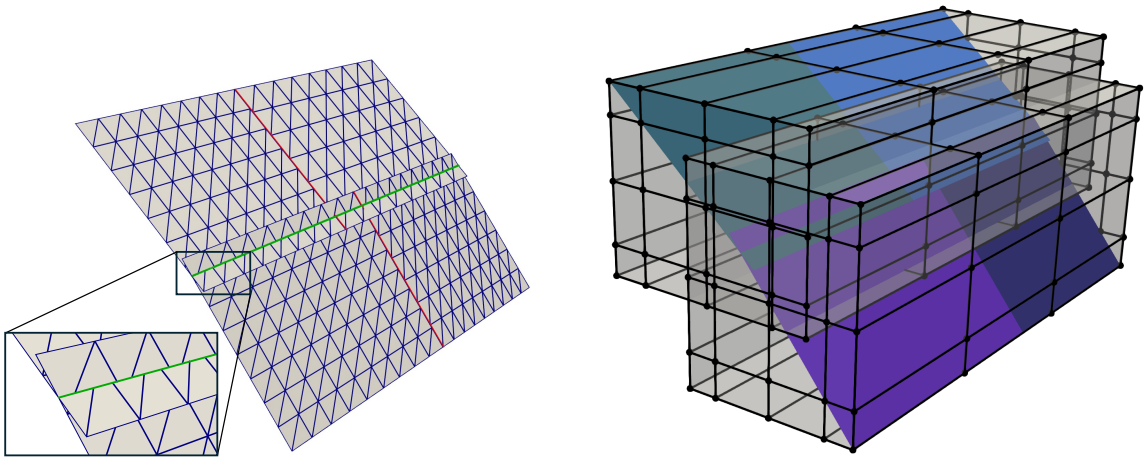
To demonstrate the effectiveness of the proposed shape optimization approach for multi-patch shell structures that incorporate a moving intersection, we test it against two T-beam benchmark problems. Both benchmarks converge to the optimal shapes with sufficiently small errors. During the optimization process, relative movement between the surface patches is achieved using analytical derivatives calculated from the adjoint method, as discussed in Section 5.1.2. Additionally, the T-junction is accurately preserved through a linear constraint applied to the parametric coordinates of the intersection's quadrature mesh.

5.3.2 Tube with follower pressure

In this section, we investigate the shape optimization of a tube subjected to an outward-facing follower unit pressure on its inner surface. We model a quarter of the tube geometry using four separately parametrized surfaces, with the initial quarter tube geometry depicted in Figure 5.10a. Symmetric boundary conditions are applied to represent the full tube. The initial tube geometry features five intersections in total, two edge–edge intersections, highlighted with red lines, and three interior–interior intersections, marked with green lines. As discussed in Section 5.2.2, we assume that the edge–edge intersections remain unchanged due to lack of relative movement, and their intersection type does not alter throughout the optimization process. On the other hand, interior–interior intersections can be moved during the shape optimization, allowing for the search of optimal intersection locations. Consequently, the upper two shell patches can move relative to the lower two patches, and the relative locations within each pair are maintained.

In this benchmark problem, we employ the FFD-based shape modification strategy, incorporating the Lagrange extraction technique [148], as introduced in [180] for automated preservation of edge–edge intersections in the upper and lower shell patch pairs. The setup of the B-spline blocks in the initial configuration are demonstrated in Figure 5.10b, where the four shell patches are distinguished by different colors. The initial quarter tube geometry ranges from 0 m to 1 m in both vertical and horizontal directions, and from 0 m to 2 m in the axial direction. Each shell patch pair is embedded in a trivariate B-spline block, with shape updates of shell patches achieved by adjusting the control points of the B-spline blocks. Due to the continuous shape modification inside the B-spline block, the edge–edge intersections are maintained. Moreover, relative movement is allowed between the distinct FFD B-spline blocks assigned to the upper and lower pairs.

In the structural analysis, we use a Young's modulus of 10^9 Pa and Poisson's ratio of 0 for the material properties of shell patches, each with a thickness of 0.01 m. Control points of each FFD block are aligned in the axial direction to ensure the tube remains straight, leading to



(a) **(b)**

Figure 5.10. (a) A quarter of the initial tube geometry consists of four non-matching cubic B-spline patches. The three interior–interior intersections are indicated with green lines, and two red lines mark the edge–edge intersections. (b) Initial configuration of the tube geometry and FFD blocks. Each set of surface patches with edge–edge intersections is embedded in one 3D B-spline FFD block to preserve the edge–edge intersection, while the interior–interior intersections between different FFD blocks are allowed to move during the shape optimization process.

the assignment of the control points in the first layer of the FFD blocks along the axial direction as design variables. In sum, there are 50 design variables in total, 25 for each FFD block. Meanwhile, the left edge of the upper FFD block and the lower edge of the lower FFD block are fixed to ensure constant positioning of the symmetric edges in the tube geometry. We employ the SNOPT optimizer with a tolerance of 10^{-2}, requiring 142 iterations to achieve convergence. Figure 5.11 displays a sequence of snapshots for the cross-sectional view of the tube geometry during the optimization process. The optimization snapshots demonstrate a gradual transition of the initial tube toward the expected circular tube. Notably, the upper pair of shell patches move freely relative to the lower pair during the optimization iterations. As the optimization progresses, the intersections between these shell pairs shift from interior positions in the initial configuration to the edges in the final configuration, eventually achieving the optimal design.

The shape of the optimized geometry is displayed in Figure 5.12a, where the red curve

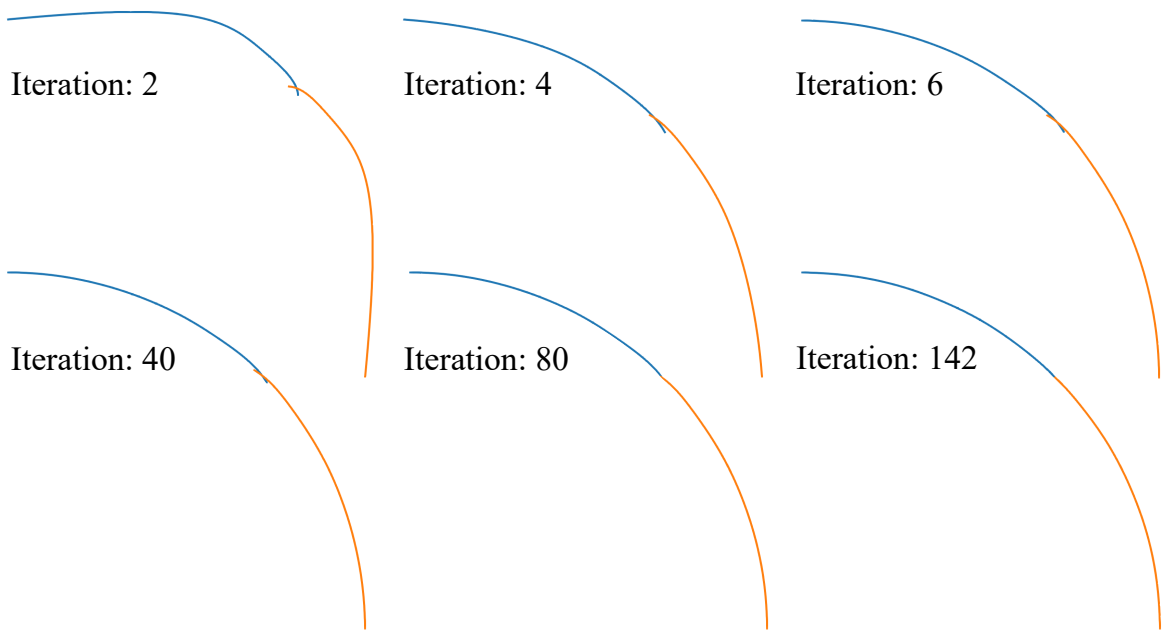
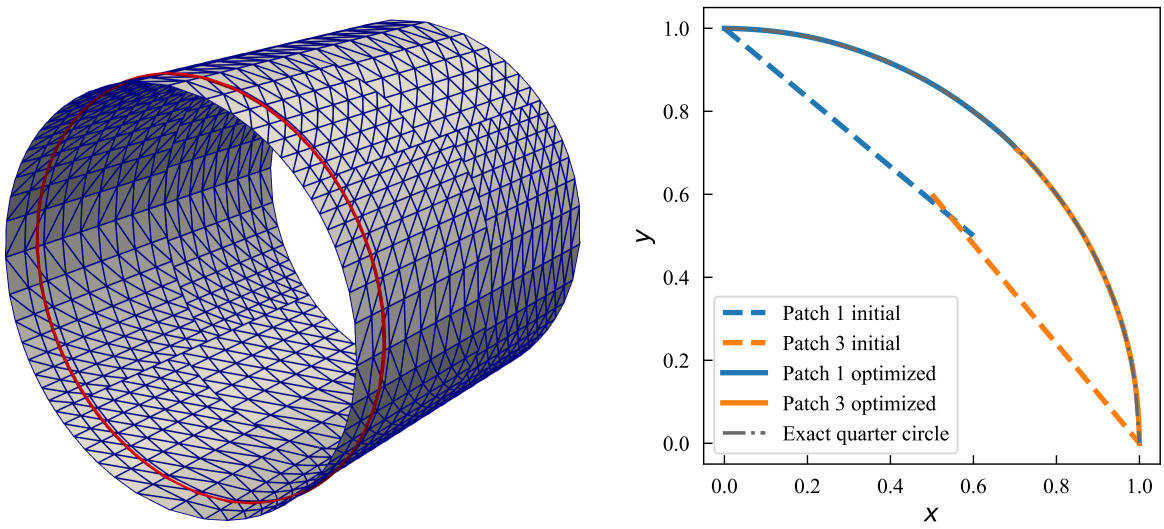


Figure 5.11. Representative snapshots of the shape optimization history for the cross-sectional view of tube geometry with interior–interior intersections. The interior–interior intersections converge to edge–edge intersections in the optimized design to minimize the internal energy of the tube.

indicates the cross-section of an exact circular tube. The circular shape represents the theoretical optimal shape that minimizes internal energy under the given follower pressure load conditions. A comparison of the cross-sectional view of the tube in initial and optimized configurations is shown in Figure 5.12b. In the optimized configuration, the cross-section of the tube geometry aligns closely with a perfect quarter circular arc, demonstrating the accuracy of the optimization approach. This tube benchmark problem highlights the capability of the optimization approach for handling intersections of the interior–interior type. This approach allows the associated intersecting shell patches to move independently, subject to a constraint guaranteeing the existence of the intersection during the shape optimization process.

Acknowledgements

A portion of this chapter has been published in “H. Zhao, J. T. Hwang, and J. S. Chen. Shape optimization of non-matching isogeometric shells with moving intersections. *Computer*



(a) **(b)**

Figure 5.12. (a) Optimized geometry of the tube, the red curve represents an exact circle for comparison. (b) Cross-sectional view of the tube geometry in the initial and optimized configurations.

Methods in Applied Mechanics and Engineering, 431:117322, 2024." The dissertation author is the primary investigator and author of this paper.

Chapter 6

Application to aircraft wings

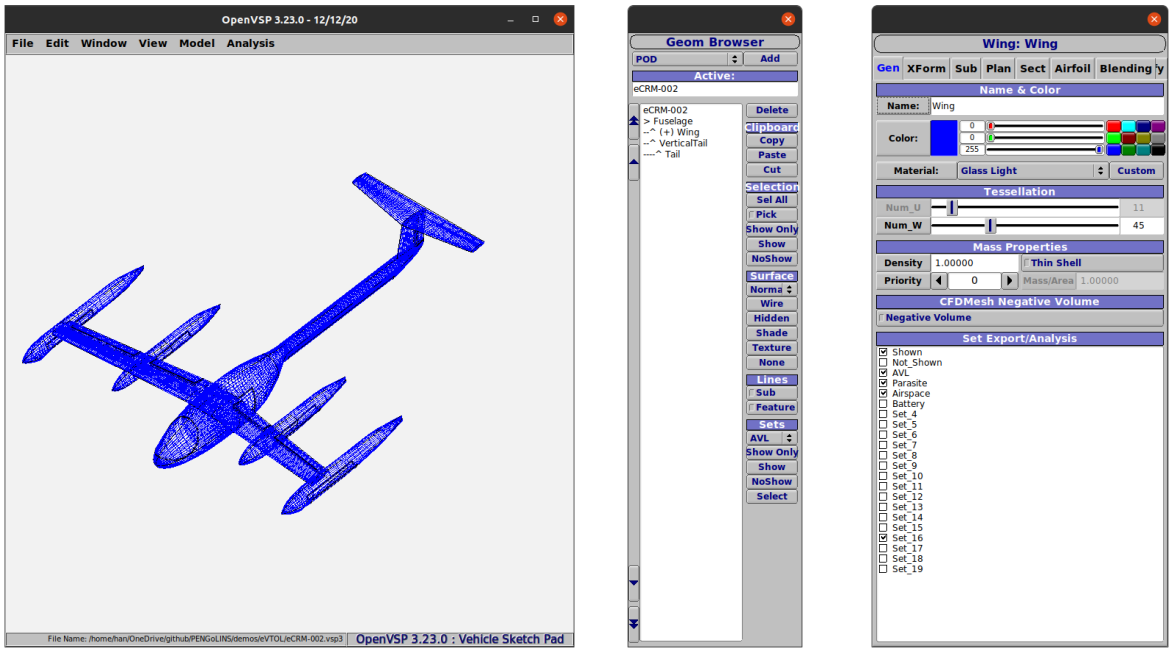
6.1 Integration of design and analysis for aerospace structures

Having verified the non-matching IGA formulation and its implementation in the open-source framework, we now turn to our target application, namely, the analysis of aerospace structures. We first discuss the source of our upstream geometry in Section 6.1.1, then apply the framework to it in Section 6.1.2, thereby demonstrating a seamless design-through-analysis workflow that entirely bypasses the problematic mesh-generation stage.

6.1.1 Design of eVTOL wing geometry

The conceptual aircraft design tool OpenVSP [61, 62, 52, 73] is the initial source of the aircraft skin geometry. It allows users to conveniently specify the geometry of major aircraft constituents, e.g., wings, fuselages, and pods, by adjusting corresponding design parameters. For instance, one can select span and chord sizes, airfoil type, location, and rotation to customize wing geometry. The VSP Hangar [160] is a public database of reference geometries provided in OpenVSP's native format, vsp3. To illustrate our design-through-analysis pipeline, we start with the eCRM-002, a common reference model (CRM) of an electric aircraft. A screenshot of the eCRM-002 geometry in OpenVSP is displayed in Figure 6.1a, alongside snapshots of the user interfaces for selecting geometry (Figure 6.1b) and adjusting design parameters for the wing

(Figure 6.1c).



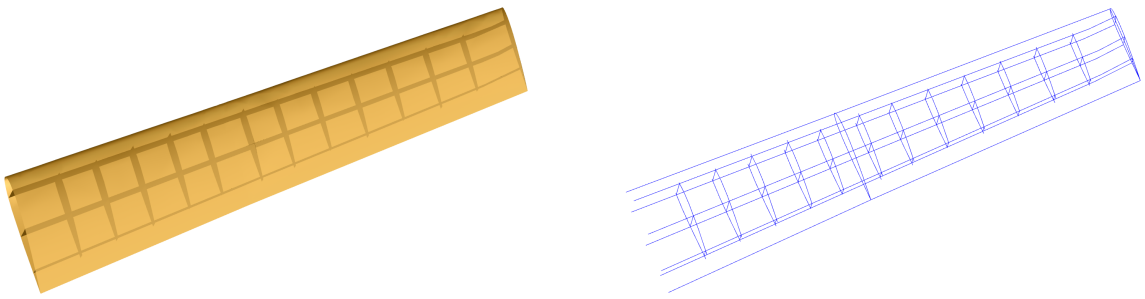
(a) **(b)** **(c)**

Figure 6.1. (a) eCRM-002 CAD model of format vsp3 in the main window of OpenVSP. (b) Geometry browser of OpenVSP for eCRM-002. (c) Aircraft wing design menu in OpenVSP.

While OpenVSP provides extensive features for designing the overall layout and external skin geometry of aircraft, it does not provide an interface for design of the internal structures typically used to stiffen airframes. In this work we use an auxiliary geometry tool to generate spline patches corresponding to the internal wing ribs and spars, to obtain a structural model sufficient for stress analysis at the conceptual design phase. This auxiliary tool provides control over the number and location of internal stiffeners, in terms of common engineering parameters (e.g., chordwise position of spars and spanwise position of ribs). It is part of a more comprehensive eVTOL geometric design framework, which remains under development, and will be described in detail by a forthcoming paper.

To perform an demonstrative stress analysis of the eCRM-002 wing, we first use OpenVSP to isolate NURBS patches corresponding to skin of one wing. We then use our auxiliary tool to introduce 3 spars and 12 ribs, modeled geometrically as NURBS patches. Given in-

dustrial CAD models, it is necessary to check the continuity of the NURBS surfaces before running the analysis, to ensure that all patches are sufficiently smooth to apply the rotation-free Kirchhoff–Love shell formulation. For example the wing model exported by OpenVSP contains excess repeated knots, leading to C^0 continuity of basis functions (despite smooth geometry); it therefore needs to be reconstructed using the method mentioned in Section 7.1.1 to obtain an analysis-suitable geometry. However, this is a straightforward procedure to apply automatically to each patch, and does not entail significant effort by an analyst. With the resulting analysis-suitable model, we can then compute non-matching intersections among the wing shell patches, as discussed in Section 7.1.1. The complete wing geometry includes 21 NURBS patches, with 87 non-matching intersections detected among them. The final analysis model of the wing, consisting of shell patches with maximal continuity and a collection of intersection curves, is rendered using Open Cascade in Figure 6.2.



(a) **(b)**

Figure 6.2. (a) eVTOL wing geometry, comprising 21 NURBS patches in total, with internal stiffeners. Upper surfaces are set translucent for visualization. (b) Computed non-matching intersections of eVTOL wing; 87 intersection curves are displayed.

6.1.2 Analysis of an eVTOL wing

In this section, we apply PENGoLINS to perform stress analysis of the analysis-suitable geometric model resulting from the design and preprocessing described in Section 6.1.1. Spline patches from this geometry are enriched via knot insertion for analysis, as shown in Figure 6.3.

The root of the wing is clamped in this analysis, and a distributed volumetric upward

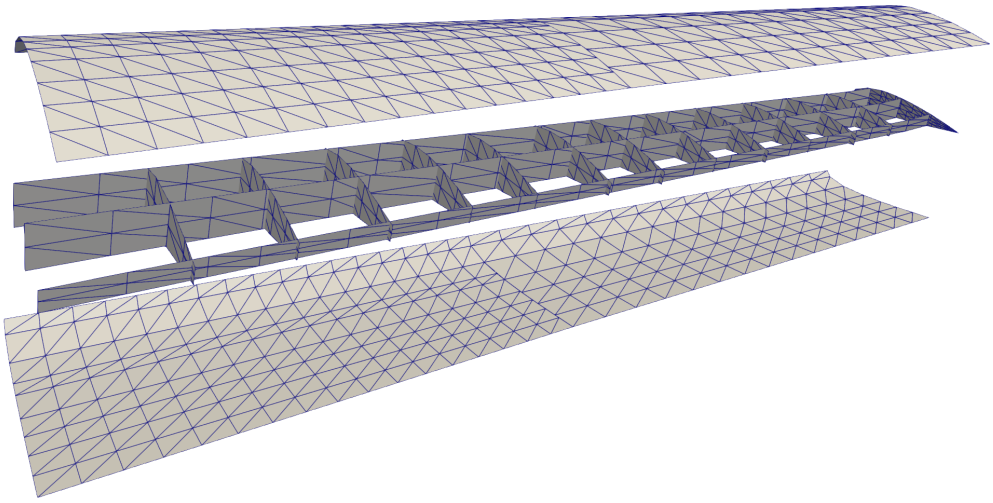


Figure 6.3. Exploded view of the eVTOL wing geometry of the eCRM-002 model, consisting of 21 NURBS shell patches with 87 intersections. The total number of displacement DoFs is 5524. Lower and upper surfaces are displaced vertically, to show the internal stiffeners.

load is applied to all shell patches, as a rough approximation of the cruising condition where one wing carries half weight of the aircraft. Aluminum is widely used in aircraft manufacturing, so we choose a Young's modulus of 68 GPa and a Poisson's ratio of 0.35.[1] The length of the wing in the spanwise direction is about 4.8 m. Its width is about 1.1 m in the chordwise direction at the root. The shell thickness is 3 mm for all patches. Assuming the take-off weight for eCRM-002 is 3000 kg, the magnitude of distributed load can be obtained by dividing through by the volume of wing (i.e., midsurface areas of patches, scaled by shell thickness). For this example, the load magnitude is determined to be 40254 N/m^3. Figure 6.4 demonstrates the stress analysis result for the geometry in Figure 6.3 with total DoFs of 5524. Smooth displacements are observed on all shell patches and the maximum von Mises stress (9.6 MPa) is located near the root, as expected.

To verify that results are converged, we perform a mesh-sensitivity study on the vertical

[1]We recognize that fiber-reinforced composite materials are a more likely choice for eVTOL aircraft, but choose an isotropic material for simplicity. The details of material modeling within each patch are largely orthogonal to the main topic of this paper, as implementing new constitutive models would be an expansion of ShNAPr, not PENGoLINS.

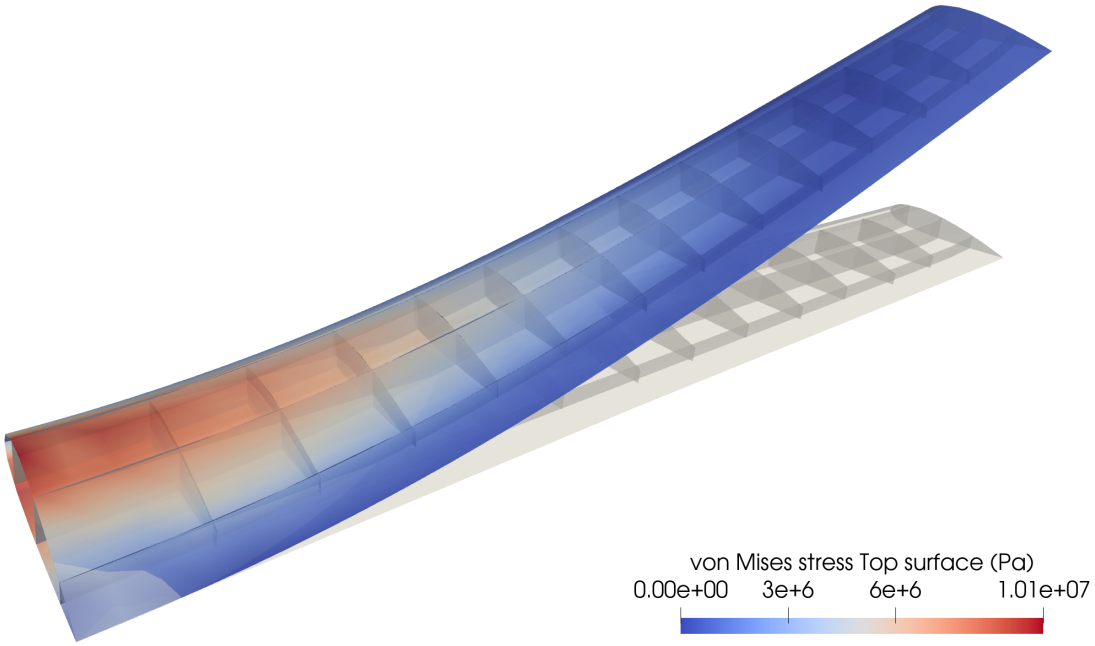


Figure 6.4. Distribution of von Mises stresses of the eVTOL wing. Wing displacements are scaled by a factor of 100, and the upper surfaces are translucent for visualization.

displacement at the wingtip on the trailing edge. This QoI is computed at several levels of refinement, and the results are plotted in Figure 6.5. A clear convergence of the measured quantity is shown in this figure, where the maximum vertical displacement on the trailing edge converges to 0.010723 m. It is worth noting that even the coarsest discretization, which simply captures the geometry, gives a value within $\sim 1.5\%$ of the converged value. This discretization error is negligible compared to the modeling error that is typically inherent to the conceptual design phase.

As an outside point of comparison, we also compute a solution using classical finite element analysis of the Reissner–Mindlin model, which is the most common approach for industrial shell analysis. In particular we use a FEniCS implementation of the Reissner–Mindlin shell element introduced by [26] (i.e., triangles with quadratic Lagrange displacements and linear Crouzeix–Raviart rotations), similar to the didactic code example provided by [22]. The vertical displacement of the wingtip trailing edge in a converged finite element analysis with 1,262,709

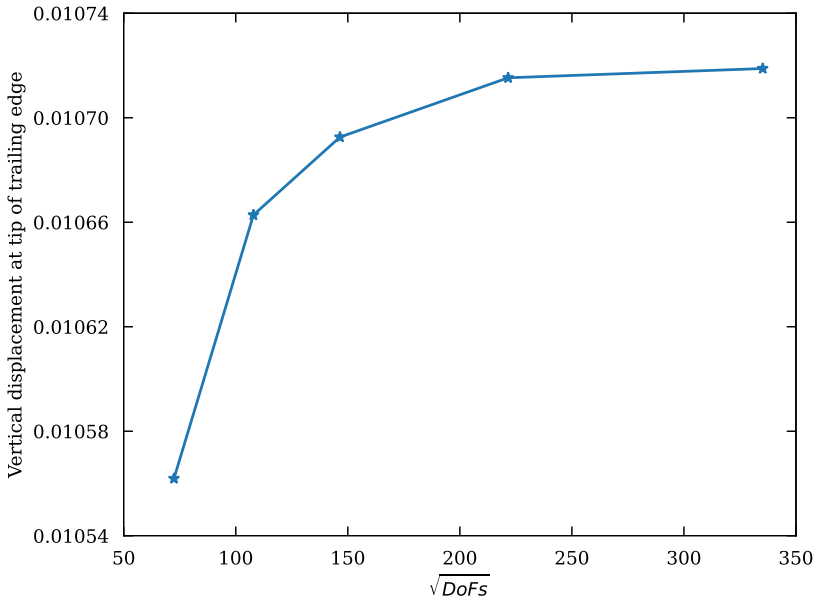
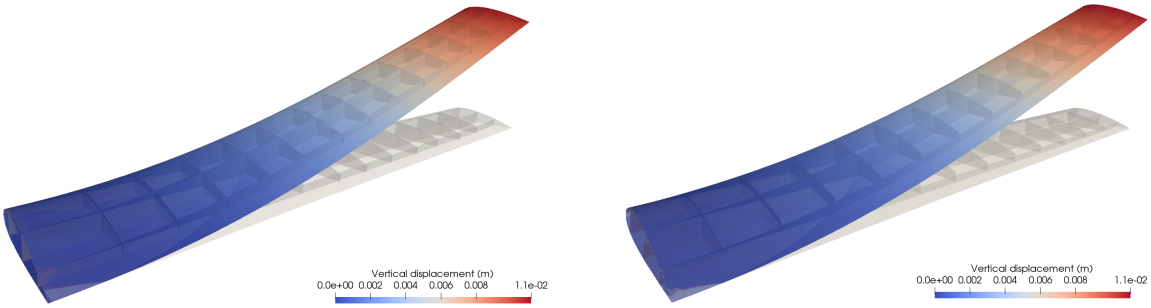


Figure 6.5. Convergence of the vertical displacement at the wingtip of the trailing edge.

DoFs is 0.010804 m.[2] We do not expect the difference between Kirchhoff–Love and Reissner–Mindlin discretizations to converge to exactly zero, because the exact solutions differ. However, the relative difference of 0.76% is well within standards of accuracy for conceptual aerospace design and comparable to the difference between converged deflections of the Scordelis–Lo roof using Kirchhoff–Love and Reissner–Mindlin models (viz. 0.3007 and 0.3024 [16, 125], respectively). The displacement solutions from PENGoLINS and classical finite element analysis are compared in Figure 6.6, and are indistinguishable for practical purposes. An important conclusion to draw from this comparison is that the Kirchhoff–Love kinematic assumptions of zero transverse shear strain and zero change in dihedral angle at creases are appropriate to the target application of aerospace structural analysis, exhibiting no meaningful loss in fidelity when compared to the prevailing industry standard of Reissner–Mindlin shell modeling.

[2]We do not claim that such a large number of DoFs is strictly necessary for accurate results with the formulation of [26], but improving per-DoF accuracy would require significant work to optimize mesh quality, which is a separate research effort beyond the scope of the present contribution (and precisely what PENGoLINS is intended to circumvent).



(a) **(b)**

Figure 6.6. (a) Displacement solution for an eVTOL wing using PENGoLINS. (b) Displacement solution using the Reissner–Mindlin shell element of [26].

6.2 PEGASUS wing thickness optimization

For the PEGASUS wing problem, we first verify the accuracy of the structural analysis using PENGoLINS for a shell structure with a large number of patches and intersections. Then two types of thickness design optimizations are performed in the following section.

6.2.1 Structural analysis of the PEGASUS wing

The PEGASUS wing CAD model is created using a customized geometry engine, an exploded view of the wing with IGA discretization is shown in Figure 6.7. The CAD model consists of 90 NURBS patches, two outer skins (one lower skin and one upper skin) and two spars (one front spar and one rear spar) connecting two adjacent ribs. The NURBS surfaces in the PEGASUS wing are represented using cubic basis functions with maximal continuity, resulting in 19572 DoFs in total. Additionally, 280 patch intersections are formed in the wing structure.

The PEGASUS wing is made of material with Young's modulus 69 GPa and Poisson's ratio 0.35, and the wing span is 12.22 m. At the wing root, the chord is 1.52 m and the airfoil thickness is 0.37 m. A uniform initial thickness is 5 mm for all patches. Considering an aircraft take-off weight of 9000 kg, a distributed upward pressure with a magnitude of 132.5 $\mathrm{N/m^2}$ is determined by dividing half of the take-off weight by the surface area of the wing. Clamped

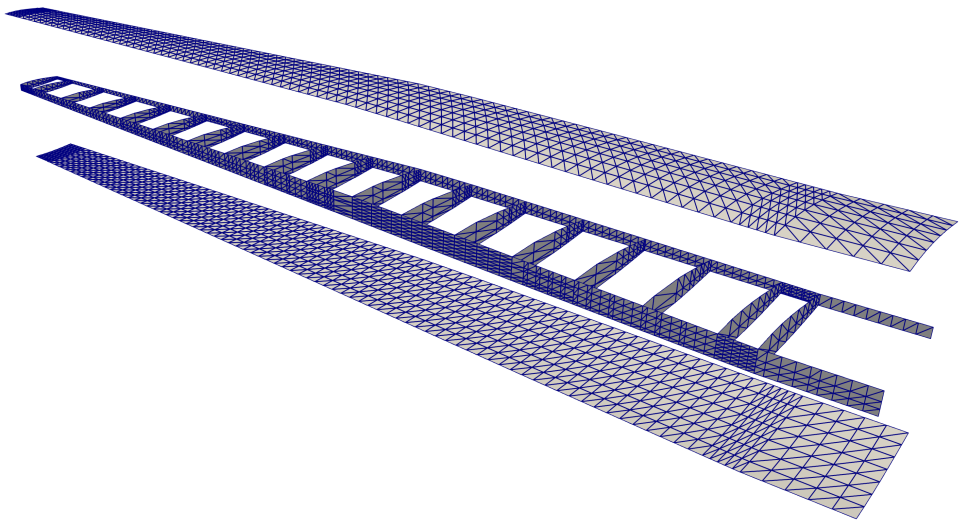
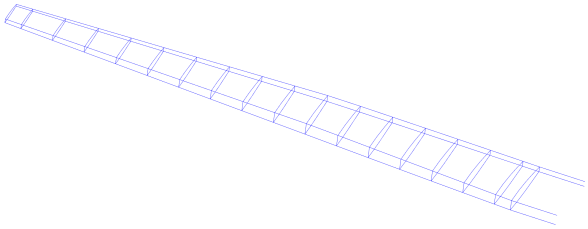


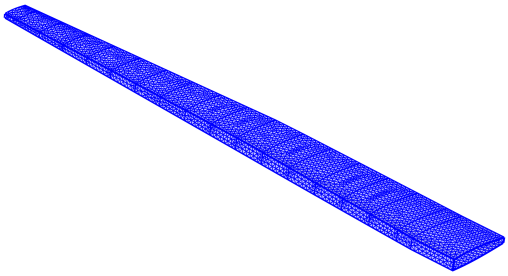
Figure 6.7. CAD geometry of the PEGASUS wing which is composed of 90 NURBS patches with 280 intersections, totaling 19572 DoFs.

boundary conditions are imposed at the wing root. Importing the PEGASUS wing geometry in IGES format to PENGoLINS, we perform structural analysis directly without finite element mesh generation. Given an analysis-suitable CAD file, the only required geometry preprocessing is to approximate surface–surface intersections, which is a much easier effort than finite element mesh generation and is automated in PENGoLINS using the functionality provided by pythonOCC. Figure 6.8a shows all the intersections presented in the PEGASUS wing, while the displacements computed by PENGoLINS are visualized in Figure 6.8c. Figure 6.8e shows the distribution of von Mises stress on top surfaces ($\xi_3 = h/2$) of the PEGASUS wing.

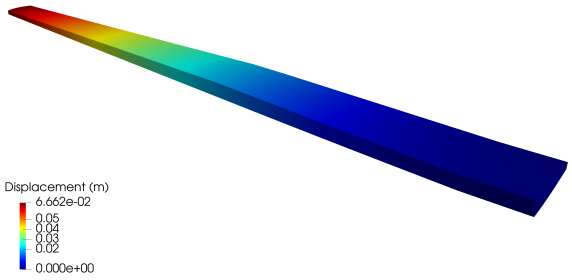
To validate the proposed non-matching coupling method for complex shell structures, we conduct a much refined FE analysis (utilizing quadratic triangular elements with 118644 DoFs) for the PEGASUS wing using COMSOL [1]. Figure 6.8b displays an extensively refined finite element mesh for the COMSOL FE analysis. Displacements solved in COMSOL and corresponding von Mises stress are depicted in Figures 6.8d and 6.8f, respectively. Figures 6.8c and 6.8d indicate that the displacements obtained from PENGoLINS closely match the results from COMSOL. The maximum displacement magnitude in PENGoLINS is 0.06662 m, which has a relative difference of 0.15% compared to the corresponding value of 0.06672 m



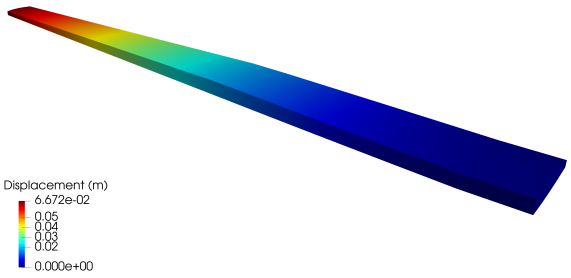
(a) Illustration of surface–surface intersections in the PEGASUS wing geometry.



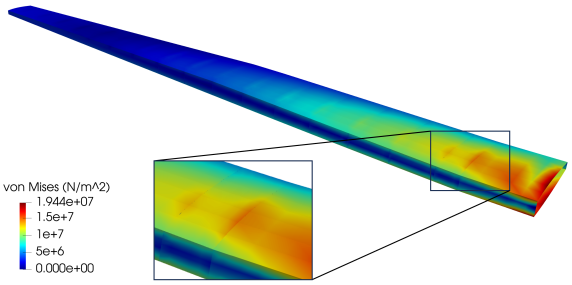
(b) Finite element mesh of PEGASUS wing generated in COMSOL.



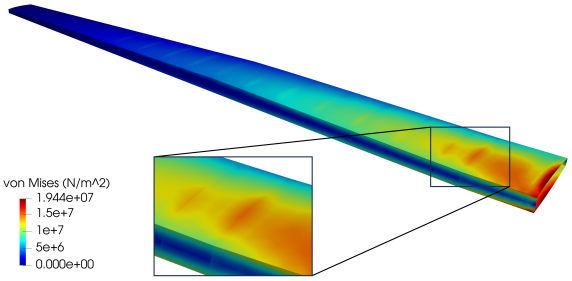
(c) Visualization of displacement magnitude solved by PENGoLINS in baseline design.



(d) Analysis result obtained from COMSOL using Reissner–Mindlin shell theory.



(e) von Mises stress on top surfaces of PEGASUS wing computed by PENGoLINS.



(f) Distribution of von Mises stress obtained from COMSOL.

Figure 6.8. Structural analysis of the PEGASUS wing using PENGoLINS, and the resulting displacement magnitude and von Mises stress are compared with the corresponding outputs obtained from COMSOL.

in COMSOL. This aligns well with the findings of a numerical experiment presented in [181, Section 5.3]. The von Mises stress distributions between PENGoLINS and COMSOL also exhibit good agreement as shown in Figures 6.8e and 6.8f, where zoom-in views are included for the observation of von Mises stress at patch intersections. Note that while Reissner–Mindlin shell

theory was employed in COMSOL, the mixed interpolation of tensorial components (MITC) [8] technique implemented in COMSOL has been carefully validated and is reliable for comparison with our solution based on Kirchhoff–Love shell theory in the limit of thin shells. The simulation results for the PEGASUS wing indicate that PENGoLINS provides good accuracy for complex shell structures, which is crucial for the subsequent design optimization.

6.2.2 Thickness optimization of the PEGASUS wing

Similar to the thickness optimization of the clamped plate discussed in Section 4.3.4, we apply the same methodology to the PEGASUS wing for piecewise constant thickness optimization. The same boundary conditions are employed throughout the optimization. In the piecewise constant thickness optimization case, a total of 90 design variables are included with lower and upper bounds of 1 mm and 100 mm, respectively. The initial thickness for all patches is taken as 5 mm. A constant volume constraint is employed in the optimization. The resulting shell thickness with minimum internal energy is depicted in Figure 6.9. The shell patch with the maximum thickness is observed at the wing root in the outer skins, while the thickness decreases consistently along the span direction for both the outer skins and spars, following a pattern similar to that of the clamped plate. The thicknesses of the internal ribs and the wingtip are close to the lower bound since the bending moments are mainly carried by the lower and upper skins given the distributed upward load. Therefore, the majority of material is redistributed towards the clamped root of the outer skins. The optimized design in Figure 6.9 gives an internal energy 38.17% less than that of the baseline configuration.

To achieve an improved design, we consider variable thickness in the outer skins and spars of the PEGASUS wing, while keeping the internal ribs with piecewise constant thickness. The configuration of the FFD blocks for variable thickness optimization is illustrated in Figure 6.10a. Four FFD blocks with quadratic bases are created to allow for variation in thickness within a single spline patch while ensuring continuity at patch intersections, with a total of 402 design variables used for this problem. By minimizing the internal energy again, the optimal

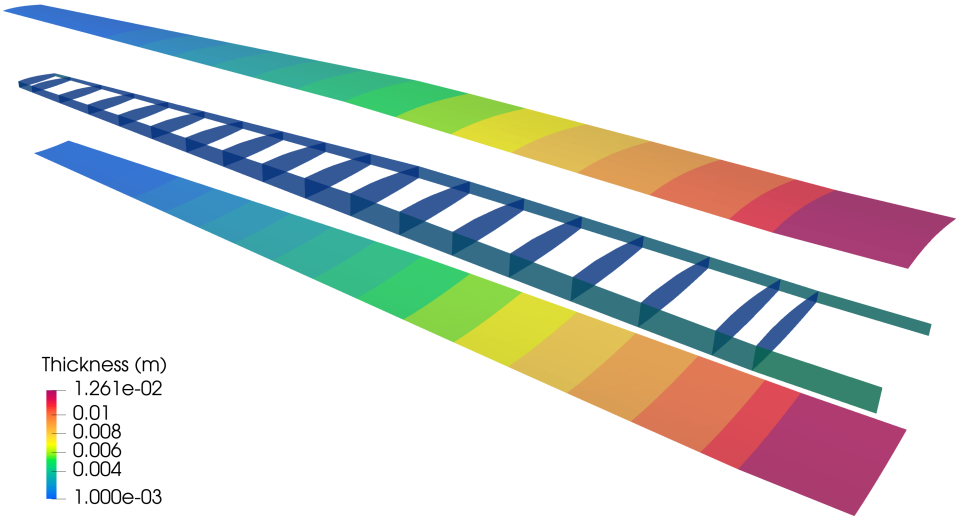
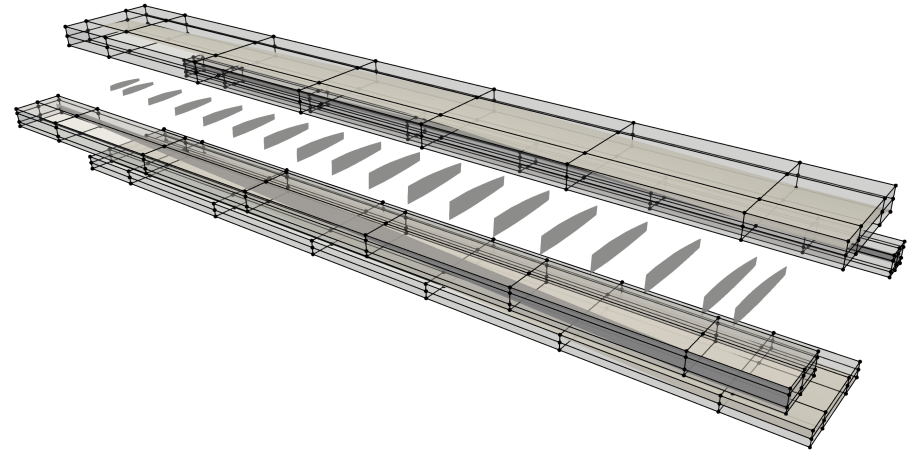


Figure 6.9. Optimization result of the PEGASUS wing with piecewise constant thickness.

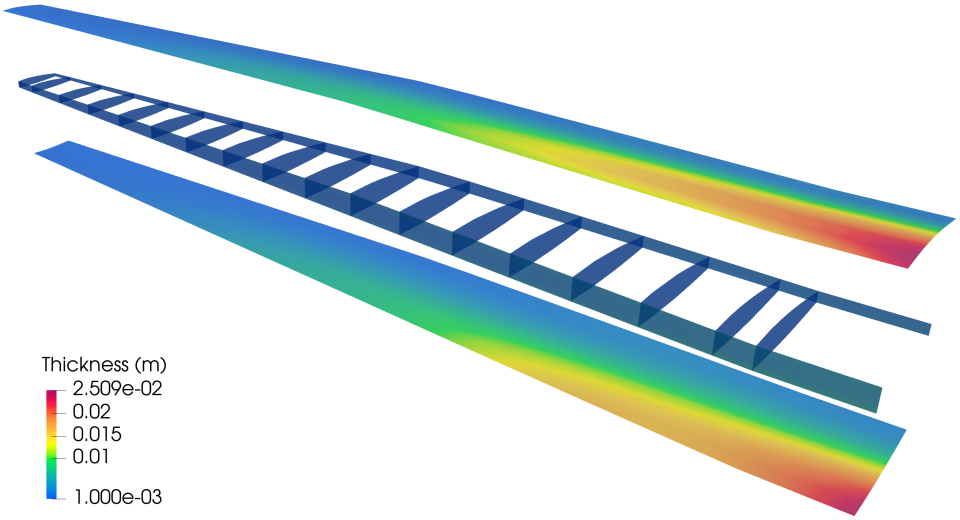
thickness distribution is obtained as shown in Figure 6.10b. Both applications in this section use the SNOPT optimizer with a tolerance of 10^{-4}. In the optimized design, the outer skins of the PEGASUS wing at the clamped edge still have the largest thickness, where the thickness distribution is smooth at the surface–surface intersections within each FFD block. The decreasing thickness trend in outer skins and spars remains consistent with the optimal piecewise constant thickness case. Comparing the combined optimization strategy with the piecewise constant method, the maximum thickness in the former is higher while maintaining the same volume. Additionally, the internal energy is reduced by 44.71% compared to the baseline design. These observations indicate that the combined thickness optimization method demonstrates a more efficient utilization of material than the piecewise constant method.

6.3 Simultaneous optimization for eVTOL wing

With the continuous advancements in aviation battery technology [119], eVTOL aircraft have emerged as a promising solution for cost-effective urban mobility [139]. In this section, we use a more advanced eVTOL wing to demonstrate the capabilities of the FFD-based optimization approach, where both the thickness and shape control points are considered as design variables



(a)

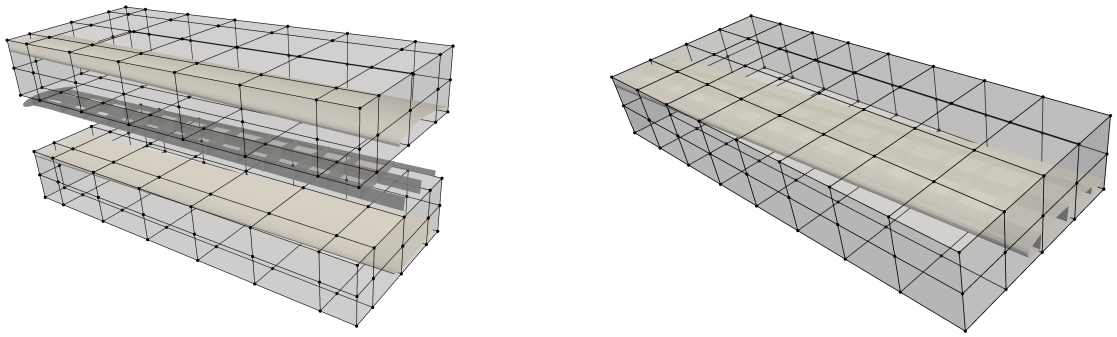


(b)

Figure 6.10. (a) Configuration of the combined thickness optimization. Each group of outer skins and spars is placed in one FFD block, and the remaining internal ribs have a piecewise constant thickness. (b) Optimal thickness distribution of PEGASUS wing.

simultaneously. By incorporating both thickness and shape coordinates in the design of shell structures, we can utilize the material more efficiently than considering thickness optimization only. The CAD geometry of the eVTOL wing, material parameters, and the corresponding structural analysis can be found in [181, Section 5]. For the optimization problem, we implement the same clamped boundary conditions and distributed upward pressure as those in the previous

application. The magnitude of pressure is set to $120\,\mathrm{N/m^2}$, approximated by dividing the take-off weight by the surface area of the eVTOL wing. The baseline configuration of the eVTOL wing, which is composed of 27 cubic NURBS patches with 87 intersections, is illustrated in Figure 6.11. We note that we perform the shape optimization for the eVTOL wing without including an aerodynamic model (which would be needed for a well-posed wing design problem) purely to provide a demonstration of the FFD-based method for complex aerospace structures.



(a) **(b)**

Figure 6.11. (a) FFD blocks for eVTOL wing thickness optimization, where the lower and upper skins have a variable thickness. Wingtips and internal ribs and spars have a piecewise constant thickness. (b) FFD block for shape optimization.

To achieve a meaningful design for the eVTOL wing, we use two quadratic B-spline FFD blocks for thickness optimization. This configuration allows for varying thicknesses in the lower and upper skins of the eVTOL, while using piecewise constant thicknesses for the internal stiffeners and wingtips. The arrangement of the thickness FFD blocks is illustrated in Figure 6.11a. Furthermore, a cubic B-spline FFD block is created for shape optimization, as depicted in Figure 6.11b. Only the vertical coordinates of control points for the shape FFD block, denoted as $\mathbf{P}_{\mathrm{FFD3}}$, are updated. In total, there are 642 design variables involved in this optimization process, and a constant volume constraint is applied as well. Regarding the thickness design variables, the lower and upper limits are selected as 1 mm and 50 mm, respectively. All shell components have initial thicknesses of 3 mm.

One challenge encountered during shape optimization of complex shell structures, such

as eVTOL wings, is the potential occurrence of oscillatory or highly distorted geometries in the updated designs. These distorted shapes can lead to poor element quality, resulting in spurious energy and affecting the accuracy of the analysis. To mitigate oscillation or radical change of shell components, an additional term is introduced in the objective function to regularize the gradient of the shape. The objective is formulated as

$$f^{\text{obj}} = \sum_{\text{I}=1}^{m} \left(W_{\text{int}}^{\text{I}} + \lambda \frac{E(\mathring{t}^{\text{I}})^3}{12\mathring{h}_A^{\text{I}}(1-\nu^2)} \int_{\mathscr{S}^{\text{I}}} \|\nabla \mathbf{P}_3^{\text{I}} - \nabla \mathring{\mathbf{P}}_3^{\text{I}}\|^2 \, d\mathscr{S} \right), \tag{6.1}$$

where I is the index of shell patches, λ is a dimensionless regularization coefficient, h_A^I is the element area of shell patch I in the physical space, $\mathbf{P}_3^{\text{I}}$ is the vertical component of the shape variable for shell patch I, and $\mathscr{S}^{\text{I}}$ is the midsurface of shell patch I. $\mathring{(\cdot)}$ denotes quantities in the baseline configuration. The regularization term in (6.1) serves as an additional artificial bending energy associated with the curvature of the eVTOL wing. Therefore, the shape oscillation can be eliminated by adjusting the regularization coefficient λ. The SNOPT optimizer is employed with a tolerance of 10^{-3}. Figure 6.12 demonstrates the optimized eVTOL wing designs achieved with varying λ values, providing insights into the influence of regularization on the final shape and thickness distribution. It can be seen that the patch intersections of the optimized shapes in Figure 6.12 are still connected using the FFD-based approach. The reductions of internal energy for different λ are listed in Table 6.1.

Table 6.1. Reduction of internal energy of the eVTOL wing after simultaneous optimization with varying regularization coefficients.

λ	1	0.1	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Internal energy reduction (%)	49.07	51.58	66.45	83.23	92.82	95.09

For this eVTOL wing optimization problem, different regularization coefficients yield distinct outcomes. A regularization coefficient of 1 and 0.1 provide optimal designs dominated by thickness update, resembling the patterns observed in the combined thickness optimization of

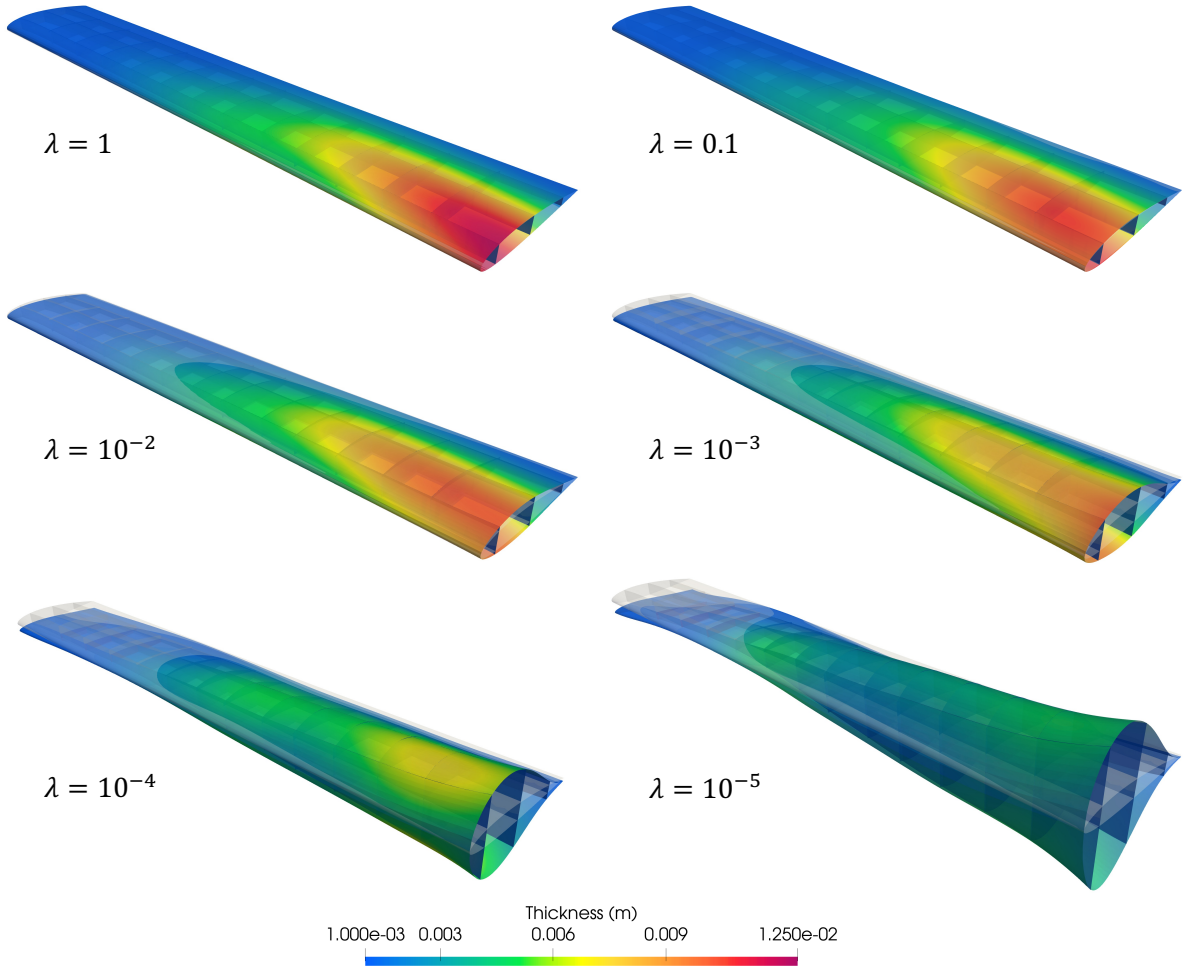


Figure 6.12. Optimized solutions of the eVTOL wing with varying regularization coefficient.

the PEGASUS wing, refer to Figure 6.10b. The optimized shapes in these two cases are similar to the baseline configuration since the artificial bending energies are the major contributor to the objective function due to the large regularization coefficients. Even slight variations in the shape variables result in substantial increases in the objective function. Conversely, small regularization coefficients, i.e., 10^{-4} and 10^{-5}, lead to considerable changes in the shape of the eVTOL wing and reduction of internal energy, amounting to 92.82% and 95.09%, respectively. However, these cases exhibit noticeable oscillations in the wingtip area, which can lead to ill-conditioned systems. On the other hand, employing regularization coefficients with values of 10^{-2} and 10^{-3} yields balanced solutions, where both the thickness redistribution and shape updates contribute to

the optimal design. The material moves towards the clamped area, accompanied by a widening of the cross-section to provide increased wing support. An exploded view of the optimal design with $\lambda = 10^{-3}$ is shown in Figure 6.13. This optimal design achieves an internal energy reduction of 83.23% compared to the baseline configuration.

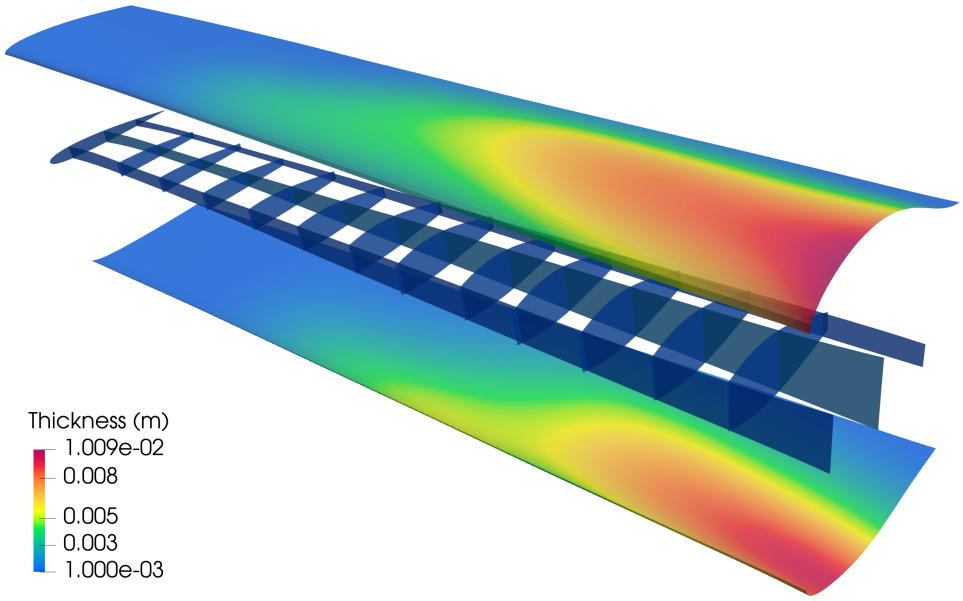


Figure 6.13. Exploded view of optimal design of eVTOL wing with regularization coefficient $\lambda = 10^{-3}$, which results in 83.23% reduction of internal energy compared to the baseline design.

6.4 Shape optimization of wing internal structures

The proposed optimization scheme holds promise for enhancing the design of novel aerospace structures, where thin-walled structures are prevalent. We apply this shell shape optimization method with moving intersections to change the internal structures layout of an eVTOL aircraft wing, aiming to reduce the internal energy of the wing. The CAD geometry of the wing is depicted in Figure 6.14, demonstrating the initial design created using the open-source software OpenVSP. The wing geometry consists of 11 B-spline patches including 2 outer skins, 1 wing tip, 2 spars, and 6 ribs, where 32 intersections are detected in the wing geometry. Among the intersections, 4 of them are categorized as edge–edge intersections between outer surfaces or the

wing tip, thus staying fixed throughout the optimization process and are marked with red curves in Figure 6.14. The remaining intersections are either interior–interior, formed between ribs and spars, or interior–edge intersections, formed between outer surfaces and internal structures, and therefore can be moved during the optimization process. These movable intersections are distinguished by green curves.

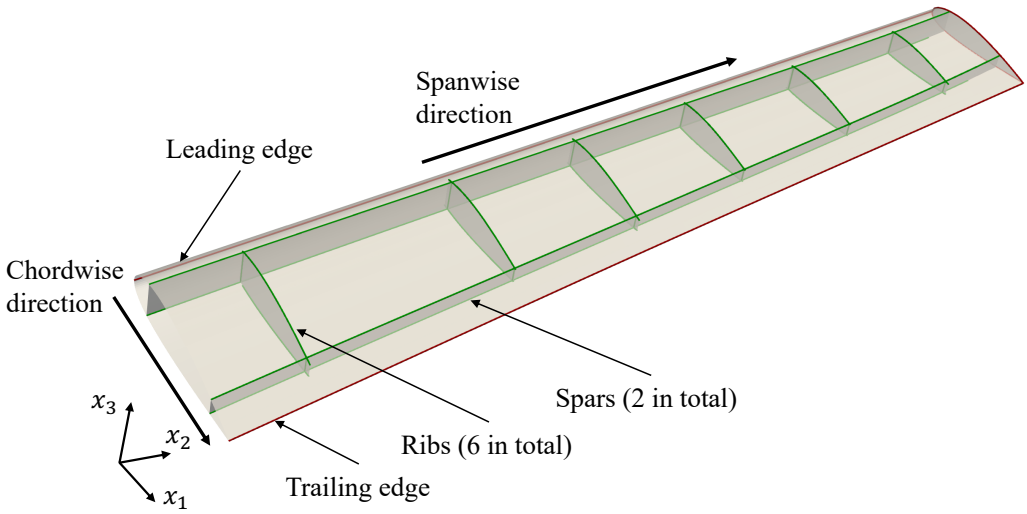


Figure 6.14. The CAD geometry of an eVTOL aircraft wing comprises 11 spline patches with 32 intersections. There are 28 movable intersections highlighted by green curves and 4 fixed intersections are indicated by red curves.

The baseline design of the wing geometry, isogeometrically discretized using cubic B-spline basis functions with a total of 2274 DoFs, is displayed in Figure 6.15, presenting the non-conforming discretizations between outer surfaces and internal structures. The wing geometry serves as the analysis model throughout the optimization process. A clamped boundary condition is applied at the wing root, while the lower outer surface is subjected to an upward distributed pressure of 500 $\mathrm{N/m^2}$, simulating the cruise condition of the wing. For structural analysis, an isotropic elastic material model is employed for simplicity, with material properties corresponding to aluminum: Young's modulus $E = 6.8 \times 10^9$ Pa and Poisson's ratio $\nu = 0.35$. The wingspan is approximately 4.8 m and the chord is 1.2 m. All shell patches have a thickness of 3 mm. The displacement field of the initial wing geometry is demonstrated in Figure 6.15.

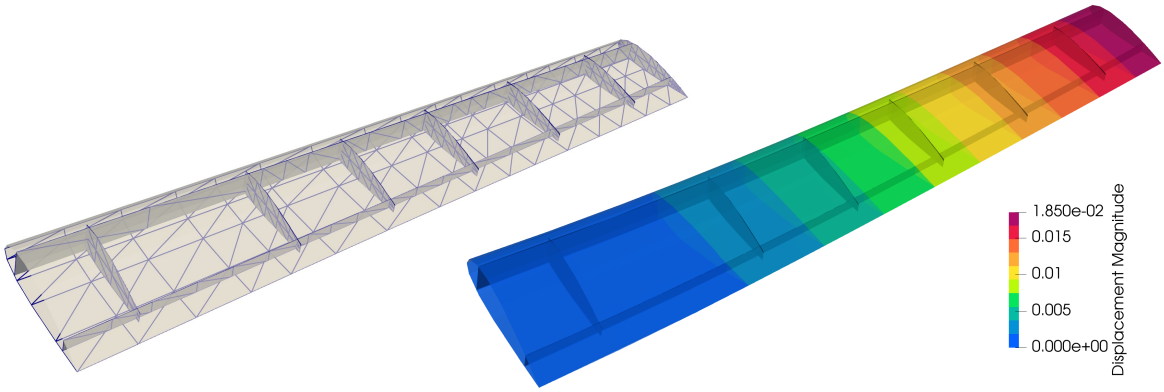


Figure 6.15. The initial eVTOL wing geometry discretization with B-spline basis functions, followed by the displacement result using the penalty-based non-matching coupling method for isogeometric Kirchhoff–Love shells. The displacement field has a unit of m and is scaled by a factor of 20 for visualization.

In this application problem, we first optimize the shape of the internal spars and ribs of the wing while keeping the outer surfaces unchanged to minimize the internal energy of the wing. For the first design scenario, we consider a rigid body approach, allowing only movement of the spars along the x_1 direction and the ribs along the x_2 direction. Thus, each spar and rib can only translate in a single direction, resulting in one associated design variable for the rigid body movement, totaling 8 design variables. However, due to the movement of the internal structures, the edges of these shell patches may deviate from the outer skins, requiring additional constraints to maintain the T-junctions. To achieve this, we utilize the constraints discussed in Section 5.2.2. Specifically, we employ a cubic spline curve with one knot span $[0,0,0,0,1,1,1,1]$ for each intersecting edge of the internal structures to preserve the T-junctions in the x_3 direction. Thus, each edge constraint requires 4 design variables. With 16 T-junctions in the wing geometry, this yields 64 design variables in the x_3 direction to the optimization problem, resulting in a total of 72 design variables for the rigid body shape optimization of the internal structures.

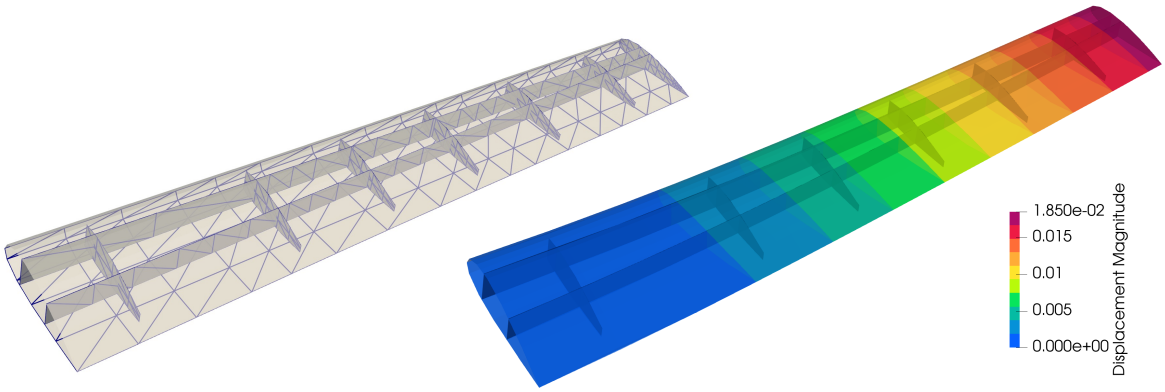
Furthermore, we introduce additional constraints to ensure that the spars remain within the envelope of the outer surfaces in the x_1 direction. This can be achieved by setting lower and upper bounds to the associated design variables. The front and rear edges of the ribs are

maintained within 80% and 15% locations from the trailing edge to the leading edge in the chordwise direction. Meanwhile, a minimum distance of 0.4 m between the two spars at the wing root and a minimum distance of 0.5 m between adjacent ribs are imposed.

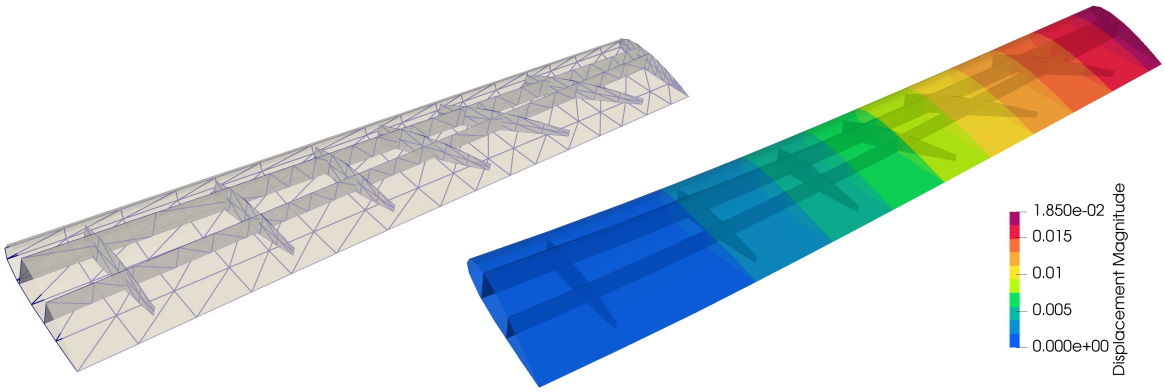
With the problem setup described above, we proceed to perform rigid body shape optimization for the internal spars and ribs. In this design scenario, we utilize the SNOPT optimizer with tolerances of 10^{-3} for optimality and 10^{-4} for feasibility. Convergence is achieved after 22 iterations, and the optimal design and associated displacement magnitude are illustrated in Figure 6.16a. In the optimized configuration, the two spars translate toward the center of the wing, reaching the minimum distance limit, while the ribs almost stay unchanged. The optimal design enhances the bending rigidity of the wing by moving the spars towards the center; a reasonable adjustment since the wing displacement is dominated by bending deformation given the geometry, loading and boundary conditions. On the other hand, the ribs, which are aligned parallel to the chordwise direction and are only allowed rigid body movement, exhibit negligible impact on reducing the bending deformation compared to the spars. The internal energy of the optimized wing geometry in Figure 6.16a is 94.98% of the baseline design in Figure 6.15.

In the second design scenario, we enable the change of the 6 ribs for not only translation but also rotation in the x_1–x_2 plane, while maintaining the planar geometry. The front and rear edges of the ribs still stay along the lines of 80% and 15% of the distance from the trailing edge to the leading edge, respectively. Thus, the x_1 coordinates of the ribs are dependent on their x_2 coordinates. This adjustment increases the number of design variables in the x_2 direction to 2 for each rib, resulting in a total of 78 design variables. To prevent excessive rotation and elongation of the ribs, an additional set of constraints is introduced to ensure that the volume of each rib remains below 1.5 times the initial volume. The minimum distance between the two spars is set as 0.1 m. The same SNOPT optimizer and convergence criteria are employed for this optimization problem. The geometry and displacement associated with the optimized configuration after 46 iterations are demonstrated in Figure 6.16b, where ribs increasingly tilt

from the wing root to the wingtip, adding additional bending rigidity to the wing, particularly in regions where the two spars are in closer proximity. The internal energy of the optimized geometry is 94.84% of the initial geometry, slightly lower than the first design scenario as we expected.



(a)



(b)

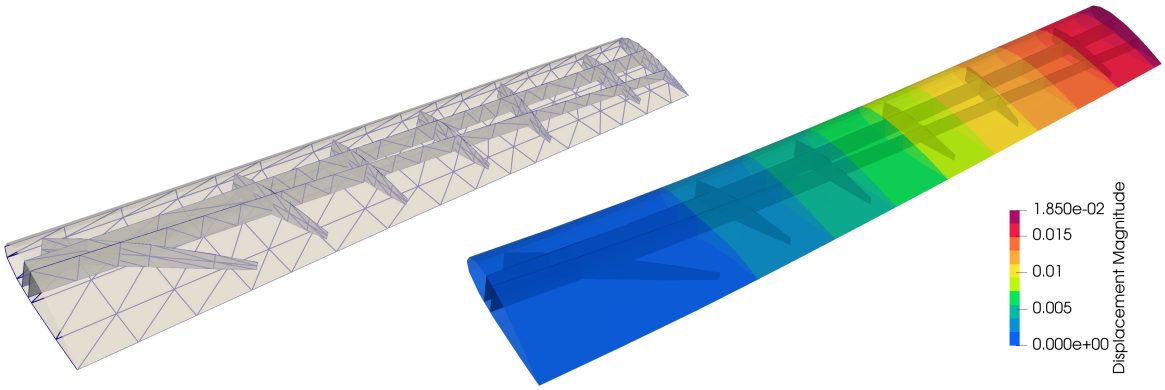
Figure 6.16. (a) The optimized geometry with rigid body translation for spars and ribs and associated displacements. (b) The optimized geometry with rigid body translation for spars and planar ribs and corresponding displacements. The displacement field is scaled by a factor of 20.

Considering that spars have more influence on the wing's bending deformation, we consider rotatable planar surfaces to model the spars in addition to the second design subjected to the same geometric and volume constraints. This enables all the internal structural members to translate and rotate in the third design scenario. Compared to the second case, one more design variable is introduced for each spar. Similar to the rib volume constraint, the volume

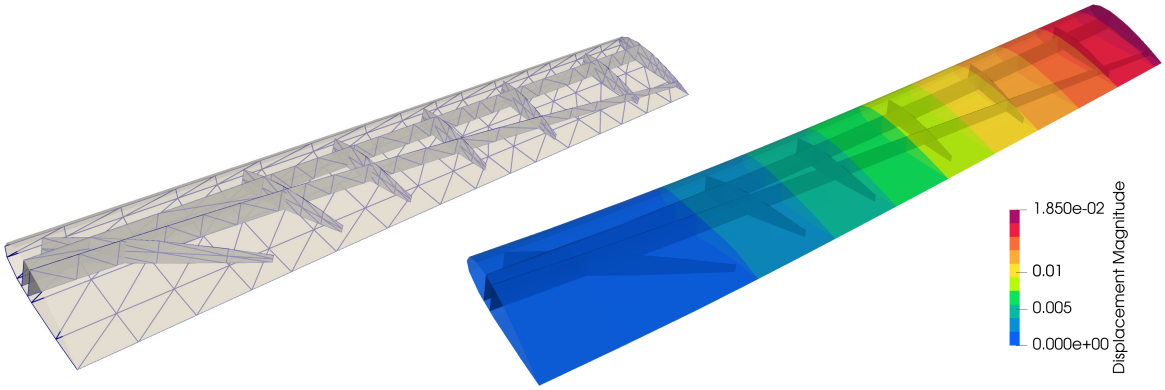
of each spar is restricted to less than 1.5 times the initial volume. All other design conditions and optimization parameters remain unchanged. Convergence is achieved after 32 iterations, yielding an optimized geometry illustrated in Figure 6.17a, where the corresponding contour plot of displacement is also included. Notably, the two spars move towards the center at the wing root and gradually split at the wingtip, while the ribs exhibit decreasing rotation from root to tip, in contrast to the second design case. Further, the results show a relation between the tilt degree of the ribs and the distance between the two spars; the closer the spars, the greater the rib tilt degrees. The internal energy of the third design is calculated to be 93.08% of the initial geometry, representing a distinct improvement compared to the second design case in Figure 6.16b.

To further improve the third design, we introduce an additional design variable to each spar in the x_1 direction so that the spars are described by quadratic B-splines. The B-splines feature a knot span of $[0,0,0,1,1,1]$ in the x_1–x_2 plane, allowing for out-of-plane curvature in addition to translation and rotation. This change increases the total number of design variables to 82. The optimizer converges in 25 iterations and yields an optimized geometry depicted in Figure 6.17b, which follows a similar pattern to the third design case but incorporates curved spars. The internal energy of the optimized design is 93.04% of the baseline design, slightly smaller than the third case. Considering other internal components in the wing and manufacturability, the third optimization scenario emerges as a more practical design, exhibiting a 6.92% internal energy reduction compared to the baseline design. However, depending on the design conditions, the fourth design provides a nontraditional internal structure that may prove beneficial for other types of stiffened thin-wall structures.

The application to the eVTOL wing demonstrates the effectiveness of the proposed approach in shape optimization of complex shell structures involving large movement of surface intersections. Wing spars and ribs are reorganized based on design criteria to minimize the internal energy of the wing. As the dimension of the design space increases, the objective function converges to a smaller value as expected. In the optimized configuration, the edges of the internal structures align well with the outer surfaces, effectively retaining the T-junctions. Throughout



(a)



(b)

Figure 6.17. (a) The optimized geometry with planar spars and ribs and contour plot of the displacements. (b) The optimized geometry with quadratic spars and planar ribs and resulting displacements. The displacement field is scaled by a factor of 20.

the optimization process, all shell patches maintain analysis-suitable NRUBS surfaces without significant distortion in the discretization despite the large movement of intersections. The eVTOL wing structural components are entirely represented by B-spline surfaces during the whole optimization process, ensuring precise shape updates in analysis and shape optimization under a streamlined workflow.

Finally, in addition to optimizing only the internal structures, considering shape optimization for both internal structures and outer skins simultaneously can achieve significant improvements for the eVTOL wing but yield a more challenging optimization problem. The internal structures must maintain the T-junctions along with the changing outer surfaces while

searching for the optimal position. In this design scenario, we consider rigid body translation for the internal ribs and spars. The design models of the lower and upper outer skins are described using surfaces with a linear B-spline in the spanwise direction and a cubic B-spline in the chordwise direction, with knot vectors $[0,0,1,1]$ and $[0,0,0,0,1,1,1,1]$, respectively. To simplify the problem, we fix the shape of the edges of each outer skin except for the edge at the wing root and only change the vertical coordinates of the outer surfaces' control points. This choice of design model introduces 8 new design variables for each outer skin surface. The upper bound of 2.9 m and lower bound of 3.3 m are set for the control points of the outer surfaces. The same boundary and loading conditions are applied to the eVTOL wing. The optimized geometry and associated displacement contour are depicted in Figure 6.18, where the internal energy of the wing is reduced by 64.54%. The wing root expands in the vertical direction, and the spars move towards the center simultaneously to enhance the bending rigidity of the wing. Meanwhile, the longer edges of the ribs and spars still maintain the T-junctions with the lower and upper surfaces despite their shape updates. Due to the use of a simple distributed load, the control points of the outer surfaces reach the specified limits to maximize the support of the wing, as expected. Aerodynamic solvers and appropriate aero-structural coupling methods are required in future work to obtain a more realistic design.

Ackonwledgements

A portion of this chapter has been published in "H. Zhao, X. Liu, A. H. Fletcher, R. Xiang, J. T. Hwang, and D. Kamensky. An open-source framework for coupling non-matching isogeometric shells with application to aerospace structures *Computers & Mathematics with Applications*, 111:109–123, 2022." The dissertation author is the primary investigator and author of this paper.

A portion of this chapter has been published in "H. Zhao, D. Kamensky, J. T. Hwang, and J. S. Chen. Automated shape and thickness optimization for non-matching isogeometric

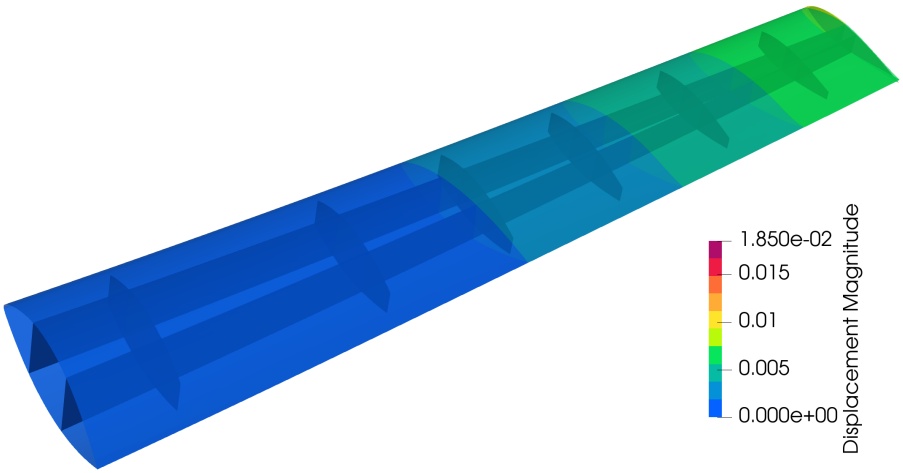


Figure 6.18. The displacement field of the optimized geometry incorporating updates of outer skins and rigid body translation of spars ribs and resulting displacements. The displacement field is scaled by a factor of 20.

shells using free-form deformation. *Engineering with Computers*, 1-24, 2024." The dissertation author is the primary investigator and author of this paper.

A portion of this chapter has been published in "H. Zhao, J. T. Hwang, and J. S. Chen. Shape optimization of non-matching isogeometric shells with moving intersections. *Computer Methods in Applied Mechanics and Engineering*, 431:117322, 2024." The dissertation author is the primary investigator and author of this paper.

Chapter 7

Open-source implementation

7.1 Analysis framework for non-matching shells

This section introduces the design and workflow of PENGoLINS, an open-source Python library that interacts with FEniCS [122] and tIGAr [98] to perform IGA for non-matching Kirchhoff–Love shells. FEniCS is a collection of software elements for automating finite element analysis. It includes the Python-based Unified Form Language (UFL) [3] for specifying variational forms, which can be automatically compiled [112] into low-level numerical routines for use with the C++ finite element library DOLFIN [123].

The library tIGAr extends FEniCS to IGA, using the concept of extraction [23, 150, 148] to represent IGA in terms of traditional finite element operations. In particular, tIGAr represents a piecewise polynomial spline space $\mathscr{V}$ through an extraction matrix $\mathbf{M}$. Each column of $\mathbf{M}$ defines a basis function from $\mathscr{V}$ by providing its coefficients in a linear combination of finite element basis functions from the Lagrange finite element space $\mathscr{V}^{\mathrm{FE}}$. Thus, linear and bilinear forms specified in UFL can be assembled using FEniCS's automated workflow to obtain the vector $\mathbf{F}^{\mathrm{FE}}$ and matrix $\mathbf{K}^{\mathrm{FE}}$, giving coordinate representations of these forms in $\mathscr{V}^{\mathrm{FE}}$. The extraction operator $\mathbf{M}$ is then used to obtain $\mathbf{F} = \mathbf{M}^{\mathrm{T}}\mathbf{F}^{\mathrm{FE}}$ and $\mathbf{K} = \mathbf{M}^{\mathrm{T}}\mathbf{K}^{\mathrm{FE}}\mathbf{M}$, to produce a linear system which can be solved for IGA degrees of freedom (DoFs). A more detailed technical explanation is given in [98].

Our starting point for tIGAr-based Kirchhoff–Love shell analysis is the existing open-

source library ShNAPr [161], which was originally developed for [97]. ShNAPr provides UFL definitions of Kirchhoff–Love kinematics and St. Venant–Kirchhoff and incompressible hyperelastic constitutive models. It also implements versatile contact mechanics using the nonlocal regularization introduced in [101]. However, ShNAPr does not include any mechanisms for enforcing displacement or rotational continuity between non-matching parts of a shell structure. PENGoLINS implements coupling at intersections between non-matching parts, while leveraging ShNAPr's UFL shell formulations to define the internal energy contributions of each part.

7.1.1 Design of PENGoLINS

The code of PENGoLINS can be grouped into three general types of functionality: preprocessing, shell coupling, and IGA. The purpose of the preprocessing module is to import CAD geometry into the analysis framework and approximate the intersection curves' coordinates in the parametric domain. Once the preprocessing stage is finished, we can proceed with computation of non-matching contributions to the PDE residual and perform IGA through extraction.

Preprocessing

The initial input to the preprocessing module is a CAD model in STEP or IGES format, consisting of a collection of untrimmed B-spline or NURBS patches. This is the geometry representation used by multiple tools for aircraft design, including the OpenVSP platform used here and GeoMACH [89], a geometry modeler for unconventional aircraft configurations. We use pythonOCC [137], a Python interface of Open Cascade with 3D modeling and CAD functionality, to locate the non-matching intersection curves between surfaces. Given two surfaces, our process for detecting intersection curves is broken into two steps, implemented within the class `BSplineSurfacesIntersections` (in PENGoLINS.occ_preprocessing):

1. First, we check for intersection curves occurring at boundaries of one or both surfaces,

using the curve–surface intersection algorithm in the pythonOCC class `GeomAPI_IntCS`.

2. If no curve–surface intersections are found in the first step, we then use the general surface–surface intersection algorithm in `GeomAPI_IntSS` to search for intersection curves that are interior to both patches.

The reason for using this two-step procedure is that the surface–surface intersection algorithm is not robust at patch boundaries, due to slight geometric imperfections in non-watertight models. This two-step procedure will fail to detect interior surface–surface intersections of patches which also have boundary–surface intersections. While mathematically possible, such cases are uncommon in models of aerospace structures, and we find the above procedure robust in practice.

Remark. *We follow the convention from pythonOCC that the class which computes surface–surface intersections only takes two surfaces. If more than two surfaces are joined together at a single intersection, we treat it as multiple intersections, and the number of intersections equals* $n = m(m-1)/2$*, where* m *is the number of surfaces joined together. By looping over all the shell patches and checking their intersections, we mark the surfaces' indices that have intersections and store them in a list, named* `mapping_list`*, for use in the computation of coupling contributions, as discussed further in Section 7.1.2.*

To perform integrals over an intersection curve between two patches, we select a discrete set of quadrature points along the curve. These quadrature points are spaced evenly in the intersection curve's parameter space. This is a low-order quadrature rule, but the quadrature error incurred by it is not significant compared to the low-order consistency error inherent in the penalty formulation. We find that results are insensitive to the density of quadrature points, so long as the spacing is smaller than the overall element size on the coupled patches. This is consistent with findings from literature on the finite cell method [147], where boundary integrals are frequently discretized irrespective of the background mesh but not found to limit convergence rates until deep into the asymptotic regime. See, e.g., [99, Section A.2.2] and [146,

Figures 5 and 6], where effectively high-order convergence is obtained with similar low-order boundary quadrature, or [86, 172], where quadrature defined on an unfitted parameterization of an immersed domain boundary is found to produce accurate results in turbulent flow analysis.

To use these quadrature points within FEniCS's automated form assembly, they must be connected into a topologically-1D mesh, which we refer to as a "quadrature mesh". The parametric coordinates of quadrature mesh quadrature points in each of two intersecting spline patches can be obtained conveniently via closest-point projection, using the Open Cascade class `GeomAPI_ProjectPointOnSurf`. The usage and methods of `BSplineSurfacesIntersections` are listed as follows:

- `__init__(surf1, surf2, tol)`: Creates a surface–surface intersection instance. `surf1` and `surf2` are instances of `Geom_Surface` in pythonOCC, and the tolerance `tol` controls the precision of the intersection curves.
- `num_intersections`: Returns the number of intersections between `surf1` and `surf2`.
- `intersections`: A list containing all computed intersection curves in the format of pythonOCC `Geom_Curve` if `num_intersections` is greater than zero. Otherwise, an empty list will be returned.
- `get_coordinates(num_pts)`: Returns a list of arrays which are the physical coordinates of the intersection curves with number of points `num_pts`.
- `get_parametric_coordinates(num_pts)`: Returns a list that contains the intersection curves' parametric coordinates with respect to `surf1` and `surf2`.

Other than intersection computation, the functionality in the preprocessing module PENGoLINS.occ_preprocessing also includes geometry manipulation to optimize the model for analysis. It is common for spline patches exported directly from CAD software to include excess repeated knots, even where the geometry is smooth. This leads to C^0 continuity of

basis functions, which violates the Kirchhoff–Love shell formulation's requirement of at least C^1 continuity. Industrial CAD geometries may also have excessive numbers of unique knots concentrated in regions without significant geometric complexity, as a byproduct of the design process (which is typically not concerned with analysis suitability). This excessive refinement leads to unnecessary degrees of freedom and can deteriorate the conditioning of algebraic systems of equations assembled during analysis. The function `reparametrize_BSpline_surface` fits analysis-suitable spline patches to the original raw CAD geometry, given a user-defined tolerance for approximation. This function uses the routine `GeomAPI_PointsToBSplineSurface` from pythonOCC to generate a new B-spline surface via a least-squares fit of points sampled from the original surface. We can then pass the resulting surface into the class `NURBSControlMesh4OCC` (in PENGoLINS.NURBS4OCC) to represent its control mesh, which is supported by tIGAr.

We wrap the aforementioned preprocessing functionality in a class `OCCPreprocessing`, which is instantiated from a user-provided CAD geometry and performs the following methods as needed:

- `reparametrize_BSpline_surfaces()`: Approximates given surfaces with desired continuity and puts the resulting surfaces in the list `self.BSpline_surfs_repara`.
- `refine_BSpline_surfaces()`: Refines B-spline surfaces through knot insertion and order elevation, then stores them in the list `self.BSpline_surfs_refine`.
- `compute_intersections()` : Computes intersections between all surfaces and generates physical and parametric coordinates of quadrature points on these intersections.

Assembling the penalty terms for one intersection curve

We now walk through the process of assembling penalty terms on the intersection curve $\mathscr{L}$ between two patches, S^{A} and S^{B}, as illustrated by Figure 7.1, in both physical space and spline parameter space. As discussed in Section 7.1.1, the preprocessing stage of the analysis creates a quadrature mesh of $\mathscr{L}$ connecting a series of quadrature points, and determines the

location of these quadrature points in the parameter spaces of S^{A} and S^{B}. We can then use the `PETScDMCollection` functionality in DOLFIN to create an interpolation matrix that transfers functions from finite element spaces on different spline patches to quadrature points on the quadrature mesh.

Remark. *Because integrals over surface–surface intersection curves are handled using the standard FEniCS machinery for form assembly, the computer algebra functionality of UFL is available to take repeated Gateaux derivatives of the penalty energy* (3.9)*, using the* `derivative` *function. Thus, only the energy* (3.9) *is directly specified in UFL; its associated virtual work and linearization are derived automatically.*

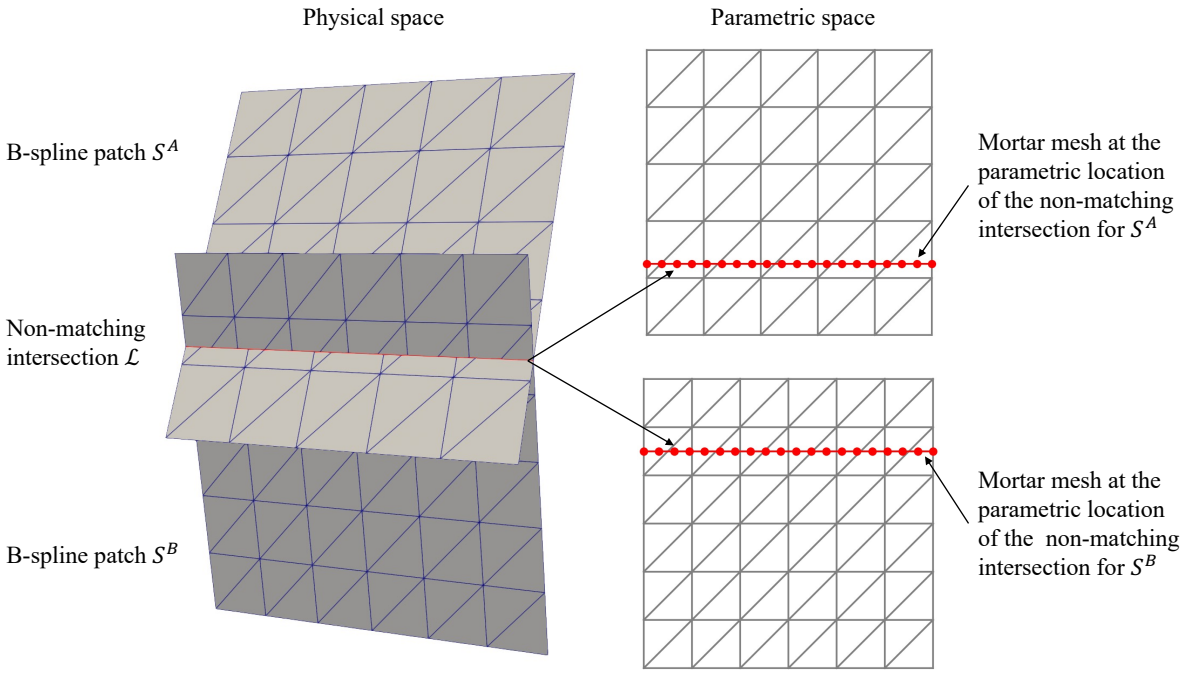


Figure 7.1. Schematic configuration of spline patches and quadrature mesh used for penalty quadrature. Patches are tesselated into triangles for technical reasons.

Remark. *Since the* α_r *term of* (3.9) *involves the first derivatives of functions from the spline patches, we must also interpolate these derivatives at quadrature points of the quadrature mesh. This involves a modified version of* `PETScDMCollection`*, which is included within PENGoLINS.*

Remark. *The* `PETScDMCollection` *functionality in DOLFIN currently only supports simplicial meshes. Thus, Bézier elements in the spline patches are each split into two triangles to perform extraction.*

For the scenario depicted in Figure 7.1, with two patches and one intersection curve, the procedure to compute non-matching coupling contributions can be summarized into the following steps:

1. Create function spaces $\mathscr{V}^{\mathrm{A,FE}}$ and $\mathscr{V}^{\mathrm{B,FE}}$ with DoFs for the finite element (FE) representation of IGA spline spaces on patches S^{A} and S^{B}.

2. Construct a quadrature mesh, $\tilde{\Omega}$, whose vertices fall on the curve $\mathscr{L}$ and act as quadrature points to approximate $\int_{\mathscr{L}}$. Further, define a function space $\mathscr{V}^{\mathrm{M}}$ with DoFs at these quadrature points.

3. Use the `GeomAPI_ProjectPointOnSurf` class from Open Cascade to project vertices of $\tilde{\Omega}$ onto S^{A} and S^{B}, and obtain parametric locations $\tilde{\Omega}^{\mathrm{M,A}}$ and $\tilde{\Omega}^{\mathrm{M,B}}$ of the vertices in the two patches.

4. Move $\tilde{\Omega}$ to configuration $\tilde{\Omega}^{\mathrm{M,A}}$ and use `PETScDMCollection` (modified as discussed in Remark 7.1.1) to generate an interpolation matrix $\mathbf{A}^{\mathrm{A}}$ from $\mathscr{V}^{\mathrm{A,FE}}$ to $\mathscr{V}^{\mathrm{M}}$. Then move $\tilde{\Omega}$ to $\tilde{\Omega}^{\mathrm{M,B}}$ and likewise generate $\mathbf{A}^{\mathrm{B}}$.[1]

5. Transfer geometrical mappings, displacements, and their parametric partial derivatives from each patch to quadrature points of the quadrature mesh. This is accomplished via left-multiplication of vectors of DoFs from $\mathscr{V}^{\mathrm{A,FE}}$ and $\mathscr{V}^{\mathrm{B,FE}}$ by interpolation matrices $\mathbf{A}^{\mathrm{A}}$ and $\mathbf{A}^{\mathrm{B}}$.

6. Use FEniCS to specify the penalty energy (3.9) in UFL and compile kernels for assembling its Gateaux derivatives (obtained automatically via the UFL `derivative` function) over

[1] As mentioned in Remark 7.1.1, we interpolate not just functions but their derivatives. The space $\mathscr{V}^{M}$ should therefore be considered a mixed space, with fields representing functions and their parametric partial derivatives, but we leave this out of the discussion for notational simplicity.

$\tilde{\Omega}$. Then assemble the vectors $\mathbf{F}^{\mathrm{A,M}}$ and $\mathbf{F}^{\mathrm{B,M}}$, corresponding to the $\mathscr{V}^M$ coordinate representations of the parts of the form $-\delta W_{\mathrm{pen}}^{\mathrm{A}B}$ that are linear in test functions from patches A and B. Likewise, assemble the matrices $\mathbf{K}^{\mathrm{AA,M}}$, $\mathbf{K}^{\mathrm{AB,M}}$, $\mathbf{K}^{\mathrm{BA,M}}$ and $\mathbf{K}^{\mathrm{BB,M}}$, corresponding to the Jacobians of $\mathbf{F}^{\mathrm{A,M}}$ and $\mathbf{F}^{\mathrm{B,M}}$ with respect to the DoFs in $\mathscr{V}^M$ used to interpolate displacements on each patch.

7. Lastly, transfer $\mathbf{F}^{\mathrm{A,M}}$ and $\mathbf{F}^{\mathrm{B,M}}$ and $\mathbf{K}^{\mathrm{AA,M}}$, $\mathbf{K}^{\mathrm{AB,M}}$, $\mathbf{K}^{\mathrm{BA,M}}$ and $\mathbf{K}^{\mathrm{BB,M}}$ to vectors and matrices corresponding to DoFs of $\mathscr{V}^{\mathrm{A,FE}}$ and $\mathscr{V}^{\mathrm{B,FE}}$. This is accomplished via matrix-vector and matrix-matrix products, using the interpolation matrices and their transposes, e.g., $\mathbf{F}^{\mathrm{A,FE}} = (\mathbf{A}^{\mathrm{A}})^{\mathrm{T}}\mathbf{F}^{\mathrm{A,M}}$, $\mathbf{K}^{\mathrm{AB,FE}} = (\mathbf{A}^{\mathrm{A}})^{\mathrm{T}}\mathbf{K}^{\mathrm{AB,M}}\mathbf{A}^{\mathrm{B}}$, etc.

The physical interpretation of the vectors and matrices, and the process of assembling a complete system of equations for the IGA spline DoFs will be clarified in Section 7.1.2.

7.1.2 Assembling the full system

This section describes the assembly of the full system of algebraic equations for a collection of intersecting patches. Consider a CAD model consisting of m isogeometric Kirchhoff–Love shells $\{S^1, S^2, \ldots, S^{\mathrm{M}}\}$ that have n non-matching intersections $\{\mathscr{L}^1, \mathscr{L}^2, \ldots, \mathscr{L}^n\}$ in the 3D physical space. A mapping list $\{(i^{11}, i^{12}), (i^{21}, i^{22}), \ldots, (i^{n1}, i^{n2})\}$, computed from preprocessing stage, contains n pairs of elements where each pair indicates the indices of two coupled shell patches, thus for any element $i^{l\alpha}$ in mapping list, $i^{l\alpha} \in \{1, \ldots, m\}$. For simplicity, we order the two indices in any pair such that $i^{l1} < i^{l2}$.

The variational problem of finding a stationary point to the combined elastic energies of the shell patches, external potentials, and penalty energies of the form (3.9) for each intersection can be written as follows: Find displacements of m shells $\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^{\mathrm{M}} \in \mathscr{V}^1, \mathscr{V}^2, \ldots, \mathscr{V}^{\mathrm{M}}$ such

that for all $\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^{\mathrm{M}} \in \mathscr{V}^1, \mathscr{V}^2, \ldots, \mathscr{V}^{\mathrm{M}}$,

$$\begin{aligned}\delta W = {} & \delta W_{\mathrm{s}}^1(\mathbf{u}^1, \mathbf{v}^1) + \delta W_{\mathrm{s}}^2(\mathbf{u}^2, \mathbf{v}^2) + \ldots + \delta W_{\mathrm{s}}^m(\mathbf{u}^m, \mathbf{v}^m) \\ & + \delta W_{\mathrm{pen}}^{i^{11} i^{12}}(\mathbf{u}^{i^{11}}, \mathbf{u}^{i^{12}}, \mathbf{v}^{i^{11}}, \mathbf{v}^{i^{12}}) + \delta W_{\mathrm{pen}}^{i^{21} i^{22}}(\mathbf{u}^{i^{21}}, \mathbf{u}^{i^{22}}, \mathbf{v}^{i^{21}}, \mathbf{v}^{i^{22}}) \\ & + \ldots + \delta W_{\mathrm{pen}}^{i^{n1} i^{n2}}(\mathbf{u}^{i^{n1}}, \mathbf{u}^{i^{n2}}, \mathbf{v}^{i^{n1}}, \mathbf{v}^{i^{n2}}) = 0 \,, \end{aligned} \tag{7.1}$$

where δW is the total virtual work of the non-matching system, δW_{s}^i is the combined internal and external virtual work of shell patch i, and $\mathscr{V}^i$ is the space of displacements on patch i.

We use $\mathbf{d}^i$ to refer to the vector of DoFs of the space $\mathscr{V}^i$. Then the variational problem (7.1) gives rise to the following system of nonlinear algebraic equations:

$$\partial_{\mathbf{d}^1} W = \mathbf{0} \,, \tag{7.2}$$

$$\partial_{\mathbf{d}^2} W = \mathbf{0} \,, \tag{7.3}$$

$$\cdots \,,$$

$$\partial_{\mathbf{d}^m} W = \mathbf{0} \,. \tag{7.4}$$

These equations are typically solved with Newton's method, or a related scheme. For linear problems, a single iteration of Newton's method can be used, starting from an initial guess of zero displacement. The linearized system of equations to solve in a single Newton step is

$$\begin{bmatrix} \partial_{\mathbf{d}^1}(\partial_{\mathbf{d}^1} W) & \partial_{\mathbf{d}^2}(\partial_{\mathbf{d}^1} W) & \ldots & \partial_{\mathbf{d}^m}(\partial_{\mathbf{d}^1} W) \\ \partial_{\mathbf{d}^1}(\partial_{\mathbf{d}^2} W) & \partial_{\mathbf{d}^2}(\partial_{\mathbf{d}^2} W) & \ldots & \partial_{\mathbf{d}^m}(\partial_{\mathbf{d}^2} W) \\ \vdots & \vdots & \ddots & \vdots \\ \partial_{\mathbf{d}^1}(\partial_{\mathbf{d}^m} W) & \partial_{\mathbf{d}^2}(\partial_{\mathbf{d}^m} W) & \ldots & \partial_{\mathbf{d}^m}(\partial_{\mathbf{d}^m} W) \end{bmatrix} \begin{bmatrix} \Delta\mathbf{d}^1 \\ \Delta\mathbf{d}^2 \\ \vdots \\ \Delta\mathbf{d}^m \end{bmatrix} = \begin{bmatrix} -\partial_{\mathbf{d}^1} W \\ -\partial_{\mathbf{d}^2} W \\ \vdots \\ -\partial_{\mathbf{d}^m} W \end{bmatrix} , \tag{7.5}$$

where the left hand side (LHS) of (7.5) is an $m \times m$ block matrix and the right hand side (RHS)

is a block vector with m entries. For the i-th block of the RHS vector,

$$\partial_{\mathbf{d}^i} W = \partial_{\mathbf{d}^i} W_{\mathrm{s}}^i + \partial_{\mathbf{d}^i} W_{\mathrm{pen}}^{ij_1} + \partial_{\mathbf{d}^i} W_{\mathrm{pen}}^{ij_2} + \ldots + \partial_{\mathbf{d}^i} W_{\mathrm{pen}}^{ij_n} \,, \tag{7.6}$$

where the superscripts of virtual work of penalty $j_1, j_2, \ldots, j_n$ are indices of n shell patches that have intersections with the i-th shell patch, and $i < j_1 < j_2 < \ldots < j_n \leq m$. For the j-th partial derivative of (7.6), which is the entry (i, j) of LHS of (7.5)

$$\partial_{\mathbf{d}^j}(\partial_{\mathbf{d}^i} W) = \begin{cases} \partial_{\mathbf{d}^i}(\partial_{\mathbf{d}^i} W_{\mathrm{s}}^i) + \partial_{\mathbf{d}^i}(\partial_{\mathbf{d}^i} W_{\mathrm{pen}}^{ij_1}) + \partial_{\mathbf{d}^i}(\partial_{\mathbf{d}^i} W_{\mathrm{pen}}^{ij_2}) + \ldots + \partial_{\mathbf{d}^i}(\partial_{\mathbf{d}^i} W_{\mathrm{pen}}^{ij_n}) \,, & j = i \\ \partial_{\mathbf{d}^j}(\partial_{\mathbf{d}^i} W_{\mathrm{pen}}^{ij}) \,, & j \in \{j_1, j_2, \ldots, j_n\} \\ \mathbf{0} \,, & \text{otherwise} \end{cases} . \tag{7.7}$$

Substituting (7.6) and (7.7) into equation (7.5), we can assemble all sub-matrices/vectors over Lagrange finite element function spaces on each patch using FEniCS's automated workflow, to obtain the system

$$\begin{bmatrix} \mathbf{K}^{11,\mathrm{FE}} & \mathbf{K}^{12,\mathrm{FE}} & \ldots & \mathbf{K}^{1m,\mathrm{FE}} \\ \mathbf{K}^{21,\mathrm{FE}} & \mathbf{K}^{22,\mathrm{FE}} & \ldots & \mathbf{K}^{2m,\mathrm{FE}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{K}^{m1,\mathrm{FE}} & \mathbf{K}^{m2,\mathrm{FE}} & \ldots & \mathbf{K}^{mm,\mathrm{FE}} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{d}^{1,\mathrm{FE}} \\ \Delta\mathbf{d}^{2,\mathrm{FE}} \\ \vdots \\ \Delta\mathbf{d}^{m,\mathrm{FE}} \end{bmatrix} = \begin{bmatrix} \mathbf{F}^{1,\mathrm{FE}} \\ \mathbf{F}^{2,\mathrm{FE}} \\ \vdots \\ \mathbf{F}^{m,\mathrm{FE}} \end{bmatrix} . \tag{7.8}$$

To obtain a system in terms of the IGA spline DoFs, we use the extraction matrix $\mathbf{M}^i$ for each

patch, in a generalization of the single-spline case summarized in Section 7.1:

$$\begin{bmatrix} \mathbf{K}^{11} & \mathbf{K}^{12} & \dots & \mathbf{K}^{1m} \\ \mathbf{K}^{21} & \mathbf{K}^{22} & \dots & \mathbf{K}^{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{K}^{m1} & \mathbf{K}^{m2} & \dots & \mathbf{K}^{mm} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{d}^1 \\ \Delta\mathbf{d}^2 \\ \vdots \\ \Delta\mathbf{d}^m \end{bmatrix} = \begin{bmatrix} \mathbf{F}^1 \\ \mathbf{F}^2 \\ \vdots \\ \mathbf{F}^m \end{bmatrix}, \tag{7.9}$$

where

$$\mathbf{K}^{ij} = \left(\mathbf{M}^i\right)^{\mathrm{T}} \mathbf{K}^{ij,\mathrm{FE}} \mathbf{M}^j \quad , \quad \mathbf{F}^i = \left(\mathbf{M}^i\right)^{\mathrm{T}} \mathbf{F}^{i,\mathrm{FE}} \tag{7.10}$$

(*without* summation on the repeated indices i and j), and $\{\Delta\mathbf{d}^i\}$ are vectors of unknown control point displacement increments for each spline patch.

The analysis procedures for non-matching isogeometric shells are wrapped into a class `NonMatchingCoupling` in the module PENGoLINS.nonmatching_coupling. The major methods of this class are:

- `__init__(splines, E, nu, h_th)`: Creates a non-matching problem instance, where the argument `splines` is a list containing m tIGAr `ExtractedSpline` objects. Material and geometric parameters `E`, `nu`, `h_th` are Young's modulus, Poisson's ratio and shell thickness, respectively. Each of them can be an instance of DOLFIN `Constant` if all shell patches share the same properties, or a list of `Constant` objects if shell patches possess different material properties.

- `create_mortar_meshes(mortar_nels, mortar_coords)`: Generates n quadrature meshes in the parametric domain, using user-defined numbers of elements `mortar_nels` and, optionally, initial mesh coordinates `mortar_coords`.

- `mortar_meshes_setup(mapping_list, mortar_parametric_coords, penalty_coefficient)`: Creates the interpolation matrices discussed in Section 7.1.1, based on the list of mappings

`mapping_list` and the quadrature mesh vertex parametric coordinates given in `mortar_parametric_coords`. The argument `penalty_coefficient` is the dimensionless penalty parameter, whose default value is 10^3.

- `set_residuals(residuals)`: Sets the residuals that correspond to the derivative of shell patches' virtual work. The argument `residuals` is a list of UFL `Form` objects, and can be obtained directly through ShNAPr. This method automatically computes the Gateaux derivatives of the provided residuals, which are then assembled to obtain blocks of the LHS matrix in (7.8). The block LHS matrix is represented using the `MatNest` abstraction from the PETSc [6, 5, 7] linear algebra backend to DOLFIN, where blocks are stored in memory as independent sparse matrix data structures.

- `solve_linear_nonmatching_problem()`: Assembles the LHS and RHS of the linear non-matching system (7.9) and solves it using either a direct or iterative solver.

7.2 Optimization framework for non-matching shells

Section 7.2.1 outlines the design of GOLDFISH and its software dependencies, which facilitate open-source implementation. A discussion of the key OpenMDAO components for shape optimization of isogeometric shell structures is presented in Section 7.2.2.

7.2.1 Software dependencies and workflow

The design of GOLDFISH leverages the code generation capabilities in FEniCS to automate the computation of symbolic Gateaux derivatives for gradient-based optimization, while OpenMDAO is used to ensure modularity and flexibility across various design conditions and disciplines. The Python library is built on a suite of open-source software dependencies, streamlining the entire design-analysis-optimization workflow.

The structural analysis is performed using PENGoLINS [181], a Python library designed for complex shell structures modeled by isogeometric Kirchhoff–Love theory. In PNGoLINS,

CAD geometries of shell structures are discretized isogeometrically and are directly available for analysis without FE mesh generation. Shell CAD geometries consisting of a collection of non-conforming NURBS surfaces are coupled using a penalty formulation [77], where a penalty energy, as discussed in (3.9), is integrated to preserve displacement continuity and angular compatibility along the intersection between shell patches. In the coupling procedure, a geometrically 2D, topologically 1D quadrature mesh is generated in the parametric space between two intersecting shell patches to serve as the integration domain for the penalty energy. This quadrature mesh is first positioned at the parametric location of the intersection with respect to the first shell patch to interpolate the displacement and covariant basis vectors. It then performs a similar operation with respect to the second shell patch. With the interpolated displacements and rotational quantities, the penalty energy and associated derivatives can be computed using FEniCS, enabling us to solve the coupled system of the complex shell structure. A schematic visualization of the coupling procedure is shown in Figure 3.2.

PENGoLINS makes use of the Python interface of OpenCASCADE [137] as the geometry engine so that CAD geometries can be imported to extract knot vectors and control points directly and use them for IGA. Another key feature inherited from OpenCASCADE is the ability to approximate NURBS patch intersections. This functionality is used to compute the parametric locations $\tilde{\boldsymbol{\xi}}$ of surface intersections, which define the integration domains for the penalty energy in (3.9). Meanwhile, the computed parametric coordinates serve as the initial guess when solving the implicit equation between NURBS surface control points and intersection parametric coordinates in (5.13). The IGA capabilities in PENGoLINS are powered by tIGAr, a Python library developed based on FEniCS that leverages the existing FE assembly routines. tIGAr constructs NURBS basis functions from Lagrange polynomials using the extraction technique [23, 148], with the Galerkin approximation process fully automated using FEniCS.

The numerical optimization is conducted using OpenMDAO [63], which computes the total derivative of optimization problems using direct or adjoint methods, depending on the problem specifications. In shape optimization problems, the number of design variables is typi-

cally much larger than the number of model outputs, such as objective functions and constraints. OpenMDAO automatically organizes the partial derivatives provided by components into total derivatives using the adjoint method, significantly improving the efficiency of derivative computation. The modular design of OpenMDAO also facilitates and standardizes the implementation of individual components for the optimization problem. Each partial derivative in (4.24) and (5.17) can be implemented as a standard OpenMDAO component, which is connected automatically during the optimization process. This modular design greatly enhances the flexibility and applicability of the code framework, allowing it to be adapted to more customized problems. A series of essential components for shell shape optimization are discussed in Section 7.2.2. The optimization problem can be solved using the open-source optimizer SLSQP [113] or the commercial optimizer SNOPT [60]. A schematic code structure of GOLDFISH is illustrated in Figure 7.2.

Figure 7.2 illustrates the streamlined workflow of GOLDFISH, where users only need to provide the NURBS-based CAD geometry of the shell structure and define the optimization problem by specifying objective functions and constraints within OpenMDAO. The code framework then automatically performs the structural analysis and shape optimization on the NURBS-based geometry without FE mesh generation. A Lagrange extraction matrix is generated for each shell patch to express the IGA spline bases by the Lagrange polynomial bases used in FEniCS. Both types of basis functions are associated with the 2D parametric meshes of the spline patches, as shown in Figure 3.2. These 2D parametric meshes are defined by the knot vectors and are generated automatically in FEniCS. The final output is also a NURBS-based geometry with updated control points that define the optimal shell shape. Throughout the optimization loop, shape updates and structural analysis are all performed directly on the CAD geometry. This integration considerably simplifies the shape optimization process.

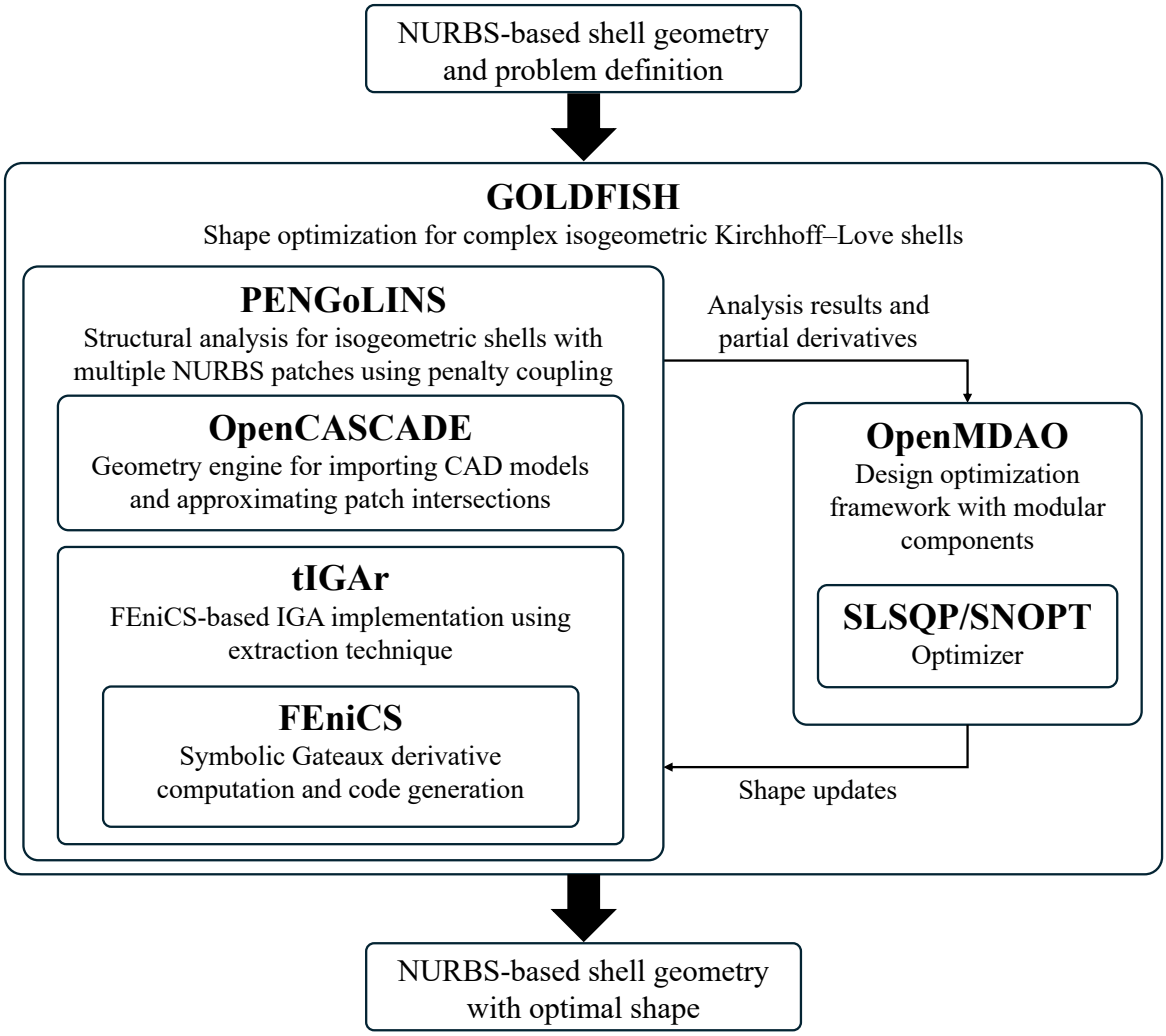


Figure 7.2. The design of Python library GOLDFISH and its software dependencies. PENGoLINS is used for structural analysis and OpenMDAO is employed for numerical optimization. Analytical partial derivatives are computed in individual components in GOLDFISH. Both input and output for the software are NURBS-based shell geometry.

7.2.2 Optimization components of shell shape optimization

This section reviews the essential building blocks for an IGA-based shape optimization problem. For the shape optimization problem described in (5.1) with the associated total derivative (5.6), the design variables are the coordinates of control points $\mathbf{P}$ of the NURBS surface defining the shell geometry. A standard InputsComp is created to provide the independent design variables to the following core components.

- *DispComp*: An implicit OpenMDAO component that takes control points $\mathbf{P}$ of the shell surface as input and returns the corresponding displacement $\mathbf{d}$ by solving the isogeometric Kirchhoff–Love shell problem $\mathbf{R}_\mathrm{S}(\mathbf{P},\mathbf{d}) = \mathbf{0}$, where $\mathbf{R}_\mathrm{S}$ is defined in (3.16). Meanwhile, this component computes partial derivatives $\partial_\mathbf{P}\mathbf{R}_\mathrm{S}$ and $\partial_\mathbf{d}\mathbf{R}_\mathrm{S}$.

- *ObjectiveComp*: An explicit OpenMDAO component calculates the objective function f for the optimization problem, which typically depends on the shape of the shell and its displacements $f(\mathbf{P},\mathbf{d})$. Additionally, this component provides partial derivatives $\partial_\mathbf{P} f$ and $\partial_\mathbf{d} f$.

With the partial derivatives computed by the two core components, the total derivative (5.6) is constructed automatically in OpenMDAO to guide shape updates until the optimal solution is reached.

7.2.3 Components for FFD-based shape optimization

As discussed in Section 4.2, the FFD-based approach is applied to real-world CAD geometries consisting of multiple non-conforming NURBS patches to maintain the intersections. In this approach, control points of the trivariate B-spline block $\mathbf{P}_\mathrm{FFD}$ serve as the design variables. Additional components are implemented for this approach to automate the optimization process, as outlined below.

- *CPFFD2SurfComp*: An explicit component computes the corresponding Lagrange nodal points $\mathbf{P}^\mathrm{FE}$ for the given control points of the FFD block $\mathbf{P}_\mathrm{FFD}$. It also provides the derivative $\mathrm{d}_{\mathbf{P}_\mathrm{FFD}}\mathbf{P}^\mathrm{FE}$ by evaluating the basis functions of the FFD block at the Lagrange nodal

points of the shell surfaces in their initial configuration.

- *CPFE2IGAComp*: An implicit component solves for the NURBS control points of the shell patches $\mathbf{P}$ given the input Lagrange nodal points $\mathbf{P}^{\text{FE}}$, using the residual equation $\mathbf{M}\mathbf{P} - \mathbf{P}^{\text{FE}} = \mathbf{0}$. The matrix $\mathbf{M}$ is the global extraction operator for the entire shell structure. This implicit component is employed to bypass the large matrix inversion as shown in (4.9). Since the degrees of freedom (DoFs) of $\mathbf{P}$ are fewer than $\mathbf{P}^{\text{FE}}$, $\mathbf{M}$ is a nonsquare matrix. The resulting $\mathbf{P}$ is interpreted as a least-squares fit to $\mathbf{P}^{\text{FE}}$.

- *DispComp*: An implicit component solves for the displacement $\mathbf{d}$ of the multi-patch shell structure given the input shell NURBS control points $\mathbf{P}$. Unlike the shape optimization of single patch shell structure, the residual becomes $\mathbf{R}(\mathbf{P}, \mathbf{d}) = \mathbf{0}$ as discussed in (3.22), which includes a penalty energy to couple the intersecting shell patches. This component also returns the partial derivatives of the non-matching residual $\partial_{\mathbf{P}}\mathbf{R}$ and $\partial_{\mathbf{d}}\mathbf{R}$.

By connecting with the previously mentioned InputsComp and ObjectiveComp, we can perform shape optimization for the complex shell structures while preserving the non-matching patch intersection throughout the optimization process. Figure 7.3 illustrates the component structure of the FFD-based shape optimization. This code structure is verified in the non-matching arch shape optimization example in Section 7.3.1.

An illustrative implementation example is provided in the following code snippets, demonstrating the use of GOLDFISH within the Python environment. First, we import the OpenMDAO and GOLDFISH libraries.

```
import openmdao.api as om
from GOLDFISH.nonmatching_opt_om import *
```

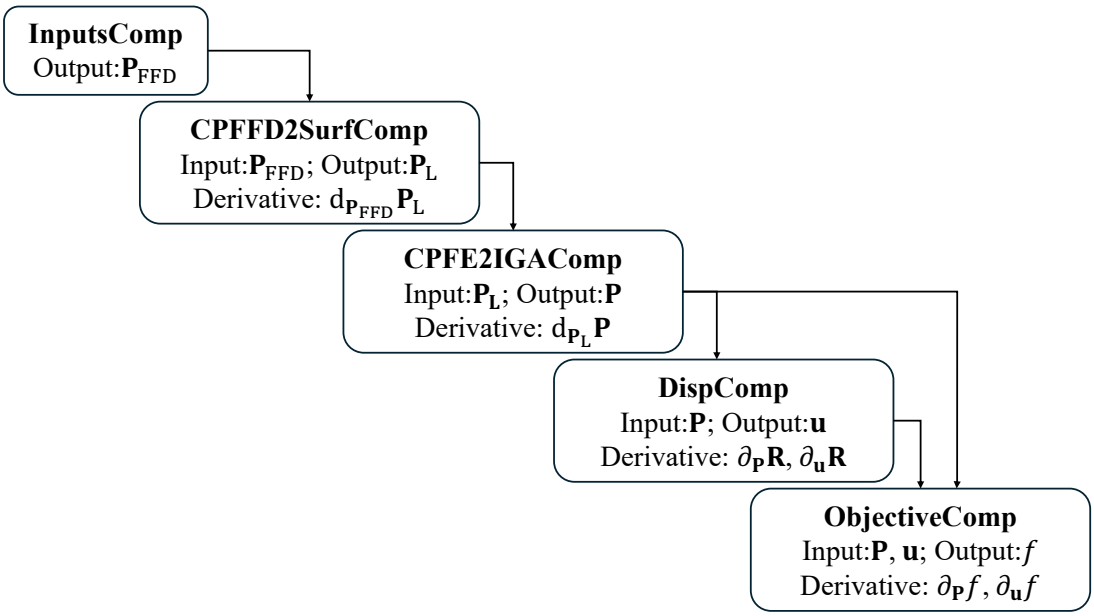


Figure 7.3. Component structure for shell shape optimization using the FFD-based approach.

Next, a class `ShapeOptGroupFFD` inherited from the OpenMDAO group is created for the FFD-based shape optimization problem, and the relevant parameters are initialized. The input of the class is an instance of the non-matching problem `NonMatchingOptFFD`, which takes the CAD geometry, analysis definitions, and optimization conditions. These problem definitions are demonstrated in Section 7.3.1.

```
class ShapeOptGroupFFD(om.Group):
    def initialize(self):
        self.options.declare('nonmatching_opt_ffd')
    # Define optimization related parameters
    def init_parameters(self):
        self.nmopt_ffd = self.options['nonmatching_opt_ffd']
        self.opt_field = self.nmopt_ffd.opt_field
        self.init_cpffd_design = self.nmopt_ffd.\
                          shopt_init_cpffd_design
        self.input_cp_shapes = [cpffd.size for cpffd
                                in self.init_cpffd_design]
```

A list of OpenMDAO components is then added to the group. The following code snippet shows

the input component which takes the control points of the FFD block as the design variables. The `CPFFD2SurfComp` and `CPFE2IGAComp` components connect control points of the FFD block to the NURBS control points of shell patches using the formulation discussed in Section 4.2.

```
    def setup(self):
        # Add inputs comp
        inputs_comp = om.IndepVarComp()
        for i, field in enumerate(self.opt_field):
            inputs_comp.add_output(
                VARNAME_CP_FFD_DESIGN+str(field),
                shape=self.input_cpffd_shapes[i],
                val=self.init_cpffd_design[i])
        self.add_subsystem('inputs_comp', inputs_comp,
                           promotes=['*'])
        # Add FFD comp
        self.ffd2surf_comp = CPFFD2SurfComp(
            nonmatching_opt_ffd=self.nmopt_ffd)
        self.ffd2surf_comp.init_parameters()
        self.add_subsystem('CPFFD2Surf_comp',
            self.ffd2surf_comp, promotes=['*'])
        # Add CPFE2IGA comp
        self.cpfe2iga_comp = CPFE2IGAComp(
            nonmatching_opt=self.nmopt_ffd)
        self.cpfe2iga_comp.init_parameters()
        self.add_subsystem('CPFE2IGA_comp',
            self.cpfe2iga_comp, promotes=['*'])
```

Furthermore, the `DispStatesComp` component performs the IGA on the updated CAD geometry and returns the structural response along with partial derivatives. In this example, we use the internal energy as the objective function, so the `IntEnergyComp` component is added to the group.

```
        # Add displacement comp
        self.disp_states_comp = DispStatesComp(
```

```
            nonmatching_opt=self.nmopt_ffd)
        self.disp_states_comp.init_parameters()
        self.add_subsystem('disp_comp',
            self.disp_states_comp, promotes=['*'])
        # Add internal energy comp (obj function)
        self.int_energy_comp = IntEnergyComp(
            nonmatching_opt=self.nmopt_ffd)
        self.int_energy_comp.init_parameters()
        self.add_subsystem('int_energy_comp',
            self.int_energy_comp, promotes=['*'])
```

Finally, we can specify the design variables and objective functions within the group to complete the setup of the optimization problem. Additionally, equality and inequality constraints can be specified in the group through the `self.add_constraint` method.

```
        # Add design variable and objective
        for i, field in enumerate(self.opt_field):
            self.add_design_var(
                VARNAME_CP_FFD_DESIGN+str(field),
                lower=DESVAR_L[i], upper=DESVAR_U[i])
        self.add_objective(VARNAME_INT_ENERGY)
```

7.2.4 Components for moving intersections

When the optimal shell structures require significant repositioning of intersections compared to the baseline design, we employ the moving intersection approach proposed in [179]. This approach allows relative movement of shell patches, ensuring that shell elements maintain good quality in the optimized geometry. In this method, we implement the multilevel design approach [128, 129] to distinguish the design model and analysis model. As such, we can select the dimension of the design space independently from the analysis model, which typically has more DoFs for accurate analysis. By selecting a design model with much fewer DoFs than the analysis model, we can expedite the convergence of the optimizer while preserving the same

geometry as the analysis model without introducing geometric errors. The multilevel design is achieved through order elevation and knot refinement of the NURBS surfaces in the design model. As a result, the design variables in this scenario are the control points of the design model. The essential components for shell shape optimization with moving intersections, combined with multilevel design, are listed below.

- *OrderElevationComp*: An explicit OpenMDAO component takes the control points of the coarse design model as input and returns corresponding control points after order elevation, following the relation $\mathbf{N}_{\mathrm{DV}}(\boldsymbol{\xi})\mathbf{P}_{\mathrm{DV}} = \mathbf{N}_{\mathrm{OE}}(\boldsymbol{\xi})\mathbf{P}_{\mathrm{OE}}$, where $\mathbf{N}_{\mathrm{OE}}(\boldsymbol{\xi})$ represents higher-order NURBS basis functions than $\mathbf{N}_{\mathrm{DV}}(\boldsymbol{\xi})$ but with the same unique knots.

- *KnotRefinementComp*: An explicit component computes control points of the analysis model $\mathbf{P}$ from the given input $\mathbf{P}_{\mathrm{OE}}$ using knot refinement for NURBS surfaces. This is achieved using similar formulation $\mathbf{N}_{\mathrm{OE}}(\boldsymbol{\xi})\mathbf{P}_{\mathrm{OE}} = \mathbf{N}(\boldsymbol{\xi})\mathbf{P}$, where $\mathbf{N}(\boldsymbol{\xi})$ has the same order as $\mathbf{N}_{\mathrm{OE}}(\boldsymbol{\xi})$ but with refined interior knots.

- *CPIGA2XiComp*: An implicit component takes control points of the analysis model $\mathbf{P}$ as inputs and solves the residual equation $\mathbf{R}_{\mathscr{L}}$ defined in (5.13) to determine the parametric coordinates of the movable intersections $\tilde{\boldsymbol{\xi}}$. Partial derivatives $\partial_{\mathbf{P}}\mathbf{R}_{\mathscr{L}}$ and $\partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R}_{\mathscr{L}}$ are computed in this component.

- *DispMintComp*: An implicit component similar to the *DispComp* in Section 5.1 that solves structural displacement $\mathbf{d}$ using the penalty-based coupling formulation for isogeometric shells. In addition to control points of the shell patches $\mathbf{P}$, this component also takes the parametric coordinates of the moving intersections $\tilde{\boldsymbol{\xi}}$ as an input to formulate the residual

$\mathbf{R}(\mathbf{P}, \tilde{\boldsymbol{\xi}}, \mathbf{d})$. This component also computes the partial derivatives $\partial_{\mathbf{P}}\mathbf{R}$, $\partial_{\tilde{\boldsymbol{\xi}}}\mathbf{R}$ and $\partial_{\mathbf{d}}\mathbf{R}$.

The total derivative in (5.17), along with a multilevel design method, can be obtained by connecting the partial derivatives provided by the components mentioned above. The connections between individual components are outlined in Figure 7.4. A numerical example verifies this code structure is demonstrated in Section 7.3.2.

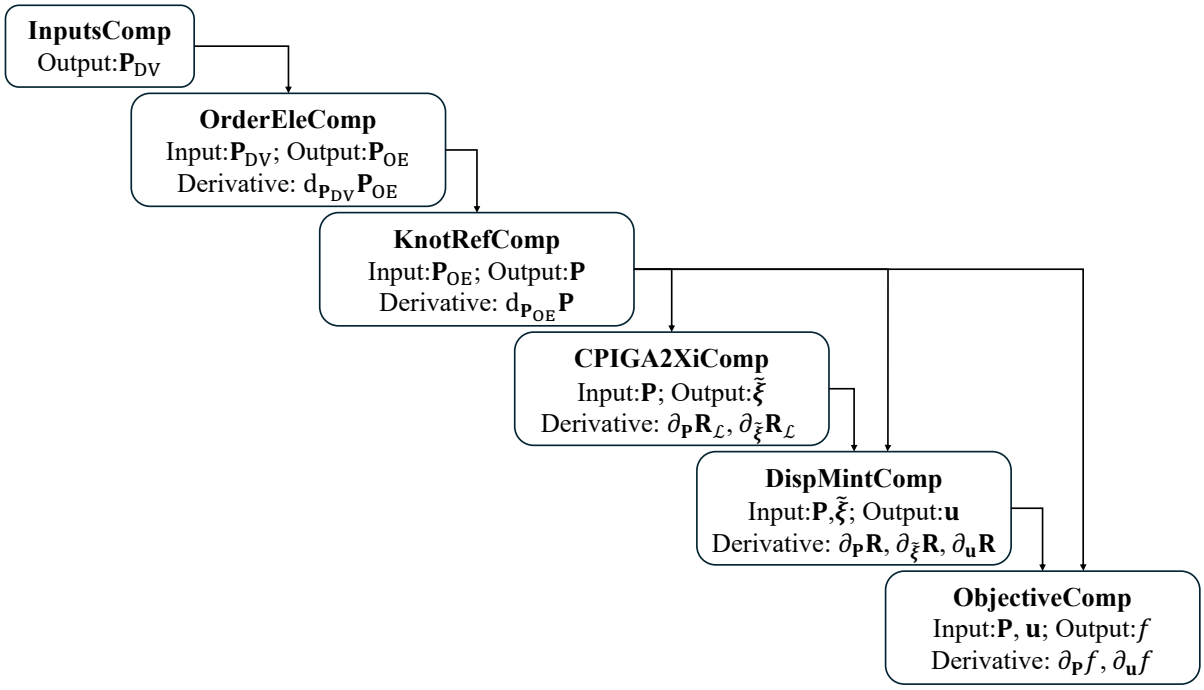


Figure 7.4. Component structure for shape optimization with moving intersections and the multilevel design method.

Code snippets are demonstrated to create the OpenMDAO group for the shape optimization of a complex shell structure with moving intersections. The class `ShapeOptGroupMint` requires two arguments. The first is an instance of `CPSurfDesign2Analysis` which provides the data and derivatives for the multilevel design of CAD geometry to reduce the dimension of the design space. The second argument is an instance of the non-matching problem `NonMatchingOpt`, which is the parent class of `NonMatchingOptFFD` without the FFD-related functions. Section 7.3.2 presents a numerical example using this group.

```
class ShapeOptGroupMint(om.Group):
    def initialize(self):
        self.options.declare('cpdesign2analysis')
        self.options.declare('nonmatching_opt')
    def init_parameters(self):
        self.des2ana = self.options['cpdesign2analysis']
        self.nm_opt = self.options['nonmatching_opt']
        self.opt_field = self.nm_opt.opt_field
        self.init_cp_design = self.des2ana.init_cp_design
        self.input_cp_shapes = [len(cp) for cp
                                in self.init_cp_design]
```

Next, we add the input component which takes control points of the coarse CAD geometry as the design variables. The order elevation and the knot refinement components are included to perform the *k*-refinement for the multilevel design, producing the fine analysis model.

```
    def setup(self):
        # Add inputs comp
        inputs_comp = om.IndepVarComp()
        for i, field in enumerate(self.opt_field):
            inputs_comp.add_output(
                VARNAME_CP_SURF_COARSE+str(field),
                shape=self.input_cp_shapes[i],
                val=self.init_cp_design[i])
        self.add_subsystem('input_comp', inputs_comp,
                           promotes=['*'])
        # Add order elevation comp
        self.cp_order_ele_comp = CPSurfOrderElevationComp(
            cpdesign2analysis=self.des2ana)
        self.cp_order_ele_comp.init_parameters()
        self.add_subsystem('CP_order_ele_comp',
            self.cp_order_ele_comp, promotes=['*'])
        # Add knot refinement comp
```

```
        self.cp_knot_refine_comp = CPSurfKnotRefinementComp(
            cpdesign2analysis=self.des2ana)
        self.cp_knot_refine_comp.init_parameters()
        self.add_subsystem('CP_knot_refine_comp',
            self.cp_knot_refine_comp, promotes=['*'])
```

Subsequently, the `CPIGA2XiComp` component is added to calculate the parametric coordinates of the intersections for a given set of surface control points by solving the implicit equation (5.13) and computing the partial derivatives. The displacement component for moving intersections `DispMintComp` is then added to evaluate the structural responses for updated surface control points and intersection locations. Similarly, the internal energy component is used to define the objective function.

```
        # Add CPIGA2Xi comp
        self.cpiga2xi_comp = CPIGA2XiComp(
            nonmatching_opt=self.nm_opt)
        self.cpiga2xi_comp.init_parameters()
        self.add_subsystem('CPIGA2xi_comp',
            self.cpiga2xi_comp, promotes=['*'])
        # Add displacement comp with moving int
        self.disp_states_comp = DispMintStatesComp(
            nonmatching_opt=self.nm_opt)
        self.disp_states_comp.init_parameters()
        self.add_subsystem('disp_comp',
            self.disp_states_comp, promotes=['*'])
        # Add internal energy comp (objective function)
        self.int_energy_comp = IntEnergyComp(
            nonmatching_opt=self.nm_opt)
        self.int_energy_comp.init_parameters()
        self.add_subsystem('int_energy_comp',
            self.int_energy_comp, promotes=['*'])
```

Lastly, we assign the coarse control points of the shell patches as design variables and designate

the internal energy as the objective function to complete the problem setup.

```
        for i, field in enumerate(self.opt_field):
            self.add_design_var(
                VARNAME_CP_SURF_COARSE+str(field),
                lower=DESVAR_L[i], upper=DESVAR_U[i])
        self.add_objective(VARNAME_INT_ENERGY)
```

The preprocessing and setup of the non-matching problems, as well as the usage of the OpenMDAO groups mentioned above, are discussed in detail in Section 7.3.

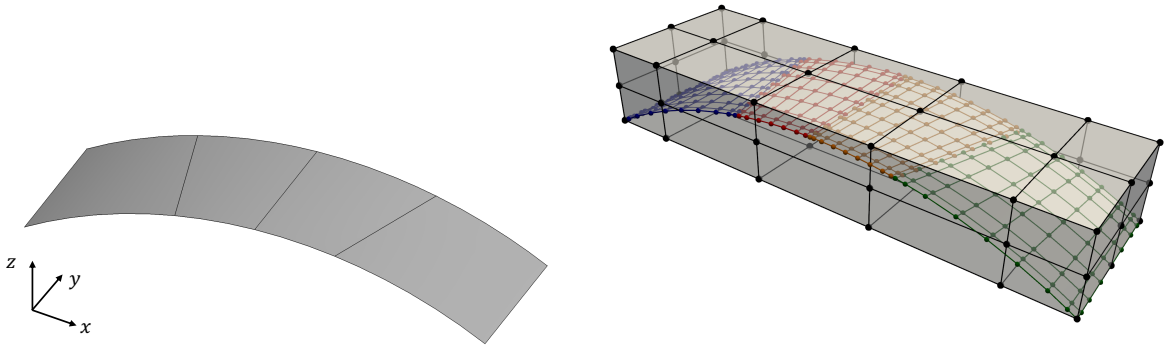
7.3 Numerical examples with code implementation

In this section, we present the implementation of benchmark problems to demonstrate the use of GOLDFISH and validate it with reference solutions. Then we showcase the application of GOLDFISH to the design optimization of aircraft wings.

7.3.1 Non-matching arch shape optimization

We use the arch shape optimization problem to verify the accuracy of the FFD-based shape update scheme in GOLDFISH. The benchmark problem was first proposed in [107, Section 8], where an arch geometry is subjected to a distributed downward load and fixed at both edges. The optimal shape of the arch with minimum internal energy is a quadratic parabola with an analytical height-to-length ratio so that the external load is entirely supported by membrane forces [107, Section 8]. To examine the capability of the FFD-based approach to preserve non-conforming surface intersections during shape updates, we create an arch geometry consisting of four non-conforming NURBS patches, as shown in Figure 7.5a. The arch geometry has a width of 3 m, a length of 10 m, and a shell thickness of 0.01 m. Young's modulus of 1000 GPa and Poisson's ratio of 0.0 are used for material properties. A 3D B-spline FFD block is generated to enclose the initial arch geometry, with the isogeometric discretization of both the FFD block and the arch geometry demonstrated in Figure 7.5b. The following listings provide essential

implementation details for the FFD-based shape optimization using GOLDFISH.



(a) (b)

Figure 7.5. (a) The initial design of an arch geometry consisting of four non-matching NURBS patches. (b) The initial arch geometry is embedded in a 3D B-spline block for FFD-based shape optimization.

We first define the material properties, coupling coefficient, and other basic parameters. Considering the three physical directions correspond to $[0, 1, 2]$ in implementation since Python is a zero-based indexing language, we select control points in the z direction for optimization and define `opt_field` as $[2]$.

```
E = Constant(1.0e12) # Young's modulus, Pa
nu = Constant(0.) # Poisson's ratio
h_th = Constant(0.01) # Shell thickness, m
pressure = Constant(1.) # Pressure magnitude, Pa
penalty_coefficient = 1.0e3 # Penalty coefficient
opt_field = [2] # Optimize z-coordinates only
ffd_block_nel = [4,1,1] # Nel of FFD block
p_ffd = 2 # Degree of the B-spline block
```

To perform the FFD-based shape optimization for the non-matching shells, we import the initial CAD geometry of the arch in IGES or STEP format into the running process using the Python interface of OpenCASCADE.

```
topo_shapes = read_igs_file("init_arch_geom.igs",
```

```
                              as_compound=False)
# Surface type conversion
occ_surf_list = [topoface2surface(face, BSpline=True)
                 for face in topo_shapes]
num_surfs = len(occ_surf_list)
```

We then create a geometry preprocessor instance to find all patch intersections and compute their parametric coordinates.

```
preproc = OCCPreprocessing(occ_surf_list)
preproc.compute_intersections(mortar_refine=2)
```

Next, we generate a list of tIGAr spline instances to build the extraction matrices. These matrices are utilized in the FFD-based shape update and IGA. The implementation of function `OCCBSpline2tIGArSpline`, which is standard to create a tIGAr spline instance from a given spline surface containing the knot vectors and control points, can be found in the GOLDFISH repository. Fixed boundary conditions are applied to the first and last shell patches and are implemented in this function. We omit these details here to focus on the setup of FFD-based shape optimization.

```
splines = []
for i in range(num_surfs):
    splines += [OCCBSpline2tIGArSpline(
                preproc.BSpline_surfs[i])]
```

The next step is to create the non-matching coupling instance for the list of tIGAr spline instances using `NonMatchingOptFFD`. This allows us to perform automated IGA and set up the FFD-based shape optimization problem.

```
nmopt_ffd = NonMatchingOptFFD(splines, E, h_th, nu)
nmopt_ffd.create_mortar_meshes(preproc.mortar_nels)
# Optimize z-coords for all shell patches
nmopt_ffd.set_shopt_surf_inds_FFD(opt_field, [0,1,2,3])
# Quadrature meshes setup for penalty energy
nmopt_ffd.mortar_meshes_setup(preproc.mapping_list,
```

```
            preproc.intersections_para_coords,
            penalty_coefficient)
```

A 3D B-spline block is created by specifying the number of elements, degrees, and limits of the control points in the three directions using function `create_3D_block`.

```
# Create the 3D B-spline FFD block
cp_lims = nonmatching_opt_ffd.cpsurf_des_lims
for field in opt_field:
    cp_range = cp_lims[field][1]-cp_lims[field][0]
    cp_lims[field][1] = cp_lims[field][1]+0.2*cp_range
FFD_block = create_3D_block(ffd_block_nel, p_ffd, cp_lims)
```

By providing the knot vectors and control points of the FFD block to the non-matching coupling instance and arranging the related constraints on the FFD block control points, we can complete the setup of the FFD block for shape optimization. The method `set_shopt_align_CPFFD` eliminates redundant design variables by aligning control points in the width direction. The `set_shopt_pin_CPFFD` method fixes control points on the lower edges of the FFD block, while `set_shopt_regu_CPFFD` ensures control points stay within the range of their adjacent neighbors, preventing unrealistic shapes.

```
# Set FFD block to the shell optimization problem
nmopt_ffd.set_shopt_FFD(FFD_block.knots, FFD_block.control)
# Set CP alignment in the width direction
nmopt_ffd.set_shopt_align_CPFFD(align_dir=[[1]])
# Set constraints to fix two lower edges of the block
nmopt_ffd.set_shopt_pin_CPFFD(pin_dir0=[2], pin_side0=[[0]],
                              pin_dir1=[1], pin_side1=[[0]])
# Set constraints to prevent self-penetration
nmopt_ffd.set_shopt_regu_CPFFD()
```

A list of the PDE residual forms based on the Kirchhoff–Love shell theory with St. Venant Kirchhoff material model is generated for all shell patches. These residual forms are expressed

as FEniCS Unified Form Language (UFL) [3] Form objects, which allows for for automatic computation of symbolic derivatives. By inputting the residual forms into `nmopt_ffd`, the stiffness matrix **K** and residual vector **R** of the non-matching shell structure are assembled in PENGoLINS subroutines.

```
source_terms = []
residuals = []
for i in range(num_surfs):
    X = nmopt_ffd.splines[i].F # Shell geometry
    # Calculate curvilinear basis vectors
    A0,A1,A2,_,_,_ = surfaceGeometry(nmopt_ffd.splines[i], X)
    v_vec = as_vector([Constant(0.), Constant(0.),
                       Constant(1.)])
    # Constant downward distributed pressure
    force = as_vector([Constant(0.), Constant(0.),
                       -pressure*inner(v_vec, A2)])
    source_terms += [inner(force, nmopt_ffd.splines[i]\
        .rationalize(nmopt_ffd.spline_test_funcs[i]))\
        *nmopt_ffd.splines[i].dx]
    residuals += [SVK_residual(nmopt_ffd.splines[i],
                  nmopt_ffd.spline_funcs[i],
                  nmopt_ffd.spline_test_funcs[i],
                  E, nu, h_th, source_terms[i])]
nmopt_ffd.set_residuals(residuals)
```

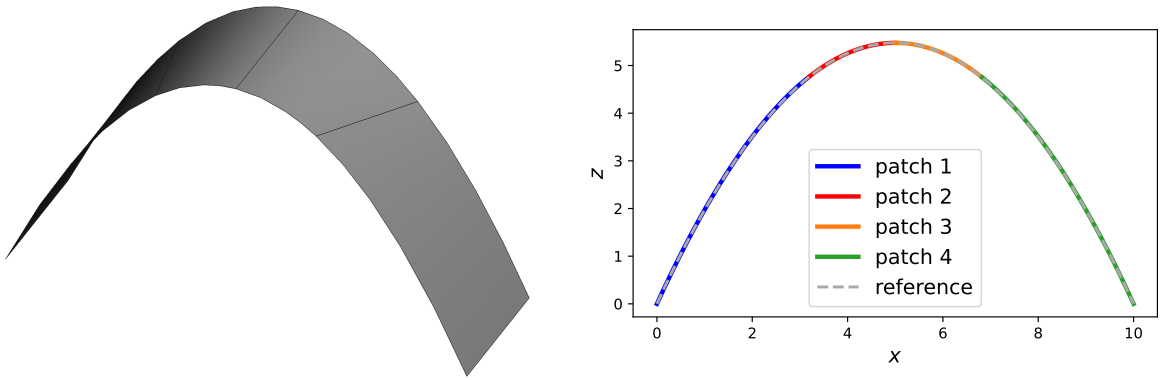
Subsequently, we can construct the FFD-based shape optimization model using the component structure and implementation discussed in Section 4.2, and then create the OpenMDAO optimization problem.

```
# Create the FFD-based shape optimization model
model = ShapeOptGroupFFD(nonmatching_opt_ffd=nmopt_ffd)
model.init_parameters()
prob = om.Problem(model=model)
```

Finally, the SLSQP optimizer with a tolerance of 10^{-12} is selected for this problem, and the objective function is minimized using method `run_driver()`.

```
prob.driver = om.ScipyOptimizeDriver()
prob.driver.options['optimizer'] = 'SLSQP'
prob.driver.options['tol'] = 1e-12
prob.driver.options['maxiter'] = 1000
# Set up and run the optimization problem
prob.setup()
prob.run_driver()
```

The optimized arch geometry after 40 iterations is shown in Figure 7.6a. A comparison of the sliced view of the optimized arch with the analytical optimal shape is presented in Figure 7.6b, where the optimized geometry closely aligns with the reference solution from [107]. The height-to-length ratio of the optimized solution is 5.4748, which shows a negligible difference from the analytical value of 5.4779.



(a) **(b)**

Figure 7.6. (a) The optimized design of the arch geometry. (b) The cross-sectional view of the optimized arch compared with the analytical optimum.

7.3.2 T-beam shape optimization with moving intersections

In this section, we use a T-beam geometry to demonstrate the use of GOLDFISH for shape optimization of multi-patch shell structures with moving intersections. The initial T-beam

geometry is described by two B-spline surfaces. The top patch is a parabolic curved surface with dimensions of 2 m in width, 5 m in length, and spans from 0 m to 0.3 m in height. The vertical patch is a flat surface with dimensions of 2 m $\times$ 5 m and is positioned at one quarter of the top patch in the horizontal direction in the baseline design. The CAD geometry of the T-beam, shown in Figure 7.7a, is subjected to a distributed pressure in the downward vertical direction and is fixed at one end.

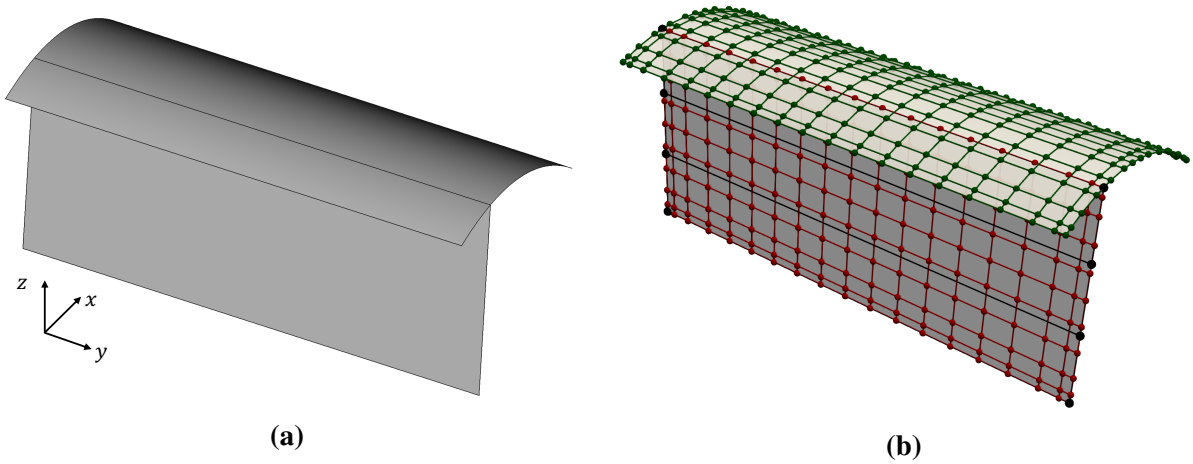


Figure 7.7. (a) The baseline design of the T-beam geometry consists of two B-spline patches. The top patch is a curved surface. (b) Isogeometric discretization of the initial T-beam geometry. The red and green control points represent DoFs in the analysis model, and the black control points represent DoFs in the design model for shape optimization.

The optimal design with minimum internal energy is obtained when the vertical patch is positioned at the center of the top patch, maintaining a constant volume. In this example, we optimize the shape of the vertical patch while keeping the geometry of the top patch fixed. Meanwhile, the intersection between the two patches is allowed to move during the optimization process. Both shell patches have a thickness of 0.1 m, with material properties of Young's modulus $E = 10^7$ Pa and Possion's ratio $\nu = 0$. The discretization of the coarse design model for the vertical patch is indicated by black lines in Figure 7.7b, while the red and green points denote the discretization of the fine analysis model. The problem setup and GOLDFISH implementation are illustrated in the following code snippets.

The problem parameters definitions and CAD geometry import are similar to the previous example and will not be repeated. For the shape optimization, an instance of the geometry processor `OCCPreprocessing` and an instance of the non-matching problem `NonMatchingOpt` are created. We specify the fields of the control points for optimization as x and z coordinates, corresponding to `opt_field` as $[0,2]$. Further, we specify the surface indices to be optimized in each field. For the vertical patch with an index of 1, the shape optimization surface indices are $[[1],[1]]$. We proceed to create and set up the quadrature meshes using `mortar_meshes_setup`, similar to the previous example. The key difference is that the argument `transfer_mat_deriv`, which defaults to 1, is set to 2 in this case since the partial derivative of the non-matching residual with respect to intersection parametric coordinates requires second-order derivations of the spline basis functions.

```
opt_field = [0,2]
shopt_surf_inds = [[1], [1]]
nmopt.set_shopt_surf_inds(opt_field, shopt_surf_inds)
nmopt.set_geom_preprocessor(preproc)
nmopt.create_mortar_meshes(preproc.mortar_nels)
nmopt.mortar_meshes_setup(preproc.mapping_list,
      preproc.intersections_para_coords,
      penalty_coefficient, transfer_mat_deriv=2)
```

We use the `check_intersections_type` method to check the types of intersections. An intersection is treated as differentiable if it is not located at the edges of both intersecting spline patches. Next, the method `create_diff_intersections` is used to generate associated data for the differentiable intersections. The argument `num_edge_pts` is a list of integers, where each item specifies the number of points in the quadrature mesh used to enforce the T-junction. For this T-beam example, there is only one T-junction, and since the vertical patch is set to remain straight in the axial direction during optimization, a single point is sufficient to ensure the T-junction.

```
preproc.check_intersections_type()
preproc.get_diff_intersections()
```

```
nmopt.create_diff_intersections(num_edge_pts=[1])
```

We then use the geometry preprocessor to initialize an instance of `CPSurfDesign2Analysis` to establish the multilevel design framework between the design model and the analysis model.

```
des2ana = CPSurfDesign2Analysis(preproc, opt_field,
                                shopt_surf_inds)
```

We assume the imported CAD geometry represents the analysis model in the workflow. Therefore, we need to define the space for the design model. We first specify the order and knots of the design model. In this example, the horizontal location of the vertical patch is described by a cubic B-spline in the ξ_1 direction and a linear B-spline in the ξ_2 direction, while the vertical location is described by linear B-splines in both directions. All spline curves in the design model have a single knot span. The orders and knot vectors for the design model are defined as follows.

```
init_p_list = [[[3,1]], [[1,1]]]
init_knots_list = [[[[0, 0, 0, 0, 1, 1, 1, 1],
                     [0, 0, 1, 1]]],
                   [[[0, 0, 1, 1],
                     [0, 0, 1, 1]]]]
```

The orders and knot vectors for the model after order elevation are given by the following variables.

```
p_list_ele = [[[3, 3]], [[3, 3]]]
knots_list_ele = [[[[0, 0, 0, 0, 1, 1, 1, 1],
                    [0, 0, 0, 0, 1, 1, 1, 1]]],
                  [[[0, 0, 0, 0, 1, 1, 1, 1],
                    [0, 0, 0, 0, 1, 1, 1, 1]]]]
```

Next, we pass the multilevel design information to the instance `des2ana`. To keep the vertical patch straight in the axial direction, the horizontal and vertical coordinates are aligned along the axial direction using the `set_cp_align` method.

```
des2ana.set_init_knots_by_field(init_p_list,
```

```
                                    init_knots_list)
des2ana.set_order_elevation_by_field(p_list_ele,
                                         knots_list_ele)
des2ana.set_knot_refinement()
des2ana.set_cp_align(field=0, align_dir_list=[1])
des2ana.set_cp_align(field=2, align_dir_list=[1])
```

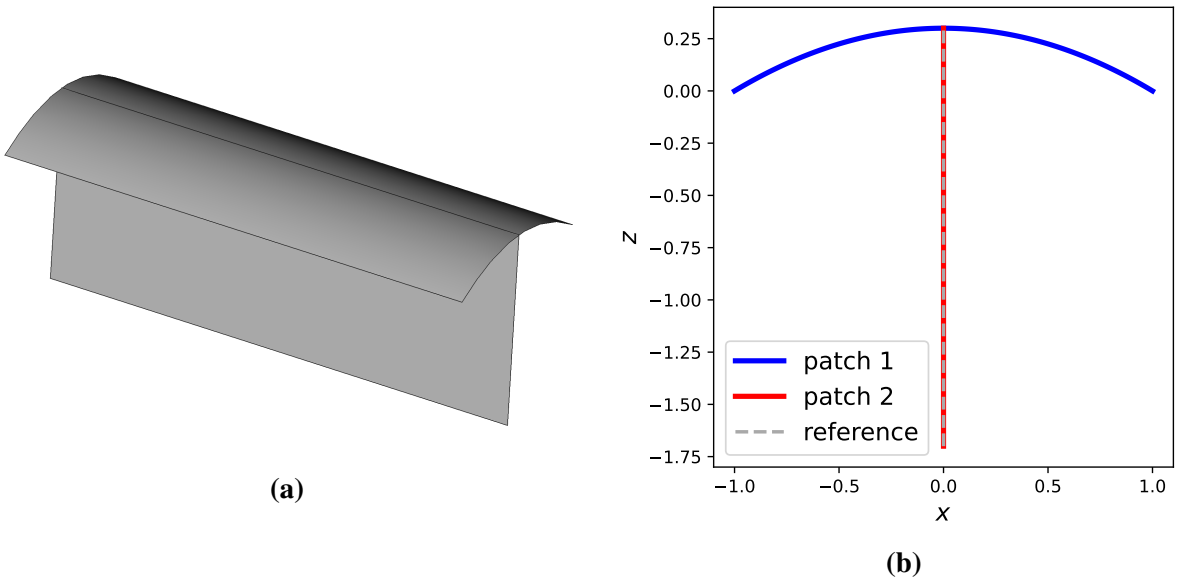
Furthermore, we can create the shape optimization model with moving intersections by passing the `nmopt` and `des2ana` instances to the `ShapeOptGroupMint` class and create the optimization problem.

```
model = ShapeOptGroupMint(cpdesign2analysis=des2ana,
                          nonmatching_opt=nmopt)
model.init_parameters()
prob = om.Problem(model=model)
```

Then the optimizer is configured using `prob.driver.options`. The optimization problem is set up with `prob.setup()` and solved by `prob.run_driver()`. Using the SLSQP optimizer with a tolerance of 10^{-9}, the optimized geometry after 22 iterations is shown in Figure 7.8a. A cross-sectional view of the T-beam is illustrated in Figure 7.8b, indicating that the vertical patch moves to the center of the top patch, thereby minimizing the internal energy of the T-beam for the given load and boundary conditions with sufficiently small errors. The optimized configuration in Figure 7.8 also demonstrates that the T-junction between the top and vertical patches is well preserved.

7.3.3 Tube under internal pressure

A tube under internal pressure is considered in this section. The initial design of the tube is shown in Figure 7.9a. To validate the GOLDFISH framework, we model a quarter of the initial tube using four non-matching B-spline patches, with three differentiable intersections between the upper and lower pair of patches. Each pair of shell patches is embedded in one FFD block to maintain their edge intersection, while the intersections between the two FFD blocks



(a) **(b)**

Figure 7.8. (a) The optimized T-beam geometry with a curved top patch. (b) Cross-sectional view of the optimized T-beam, the vertical patch is moved to the center of the top patch and maintains the T-junction.

are allowed to move during the optimization process. This setup enables the analytical optimal solution, which is a cylindrical tube. The configuration of the FFD blocks and their associated discretizations are illustrated in Figure 7.9b. The optimization of this tube combines both the FFD-based approach and the moving intersections method.

The following code snippets illustrate the setup of the tube optimization problem using GOLDFISH. As in previous examples, we begin by creating instances of `OCCPreprocessing` and `NonmatchingOptFFD` for the imported CAD geometry. For both FFD blocks, we optimize the vertical and horizontal coordinates of their control points. Since the axial direction of the tube is the z direction, the optimization field is set as [0,1] for both FFD blocks. Shell patches with indices 0 and 1 are embedded in the first FFD block, while the remaining two patches are embedded in the second FFD block. Thus, the `opt_surf_inds` is defined as [[0,1], [2,3]]. This information is passed to the non-matching problem using the method `set_shopt_surf_inds_multiFFD`.

```
opt_field = [[0,1],[0,1]]
opt_surf_inds = [[0,1], [2,3]]
```

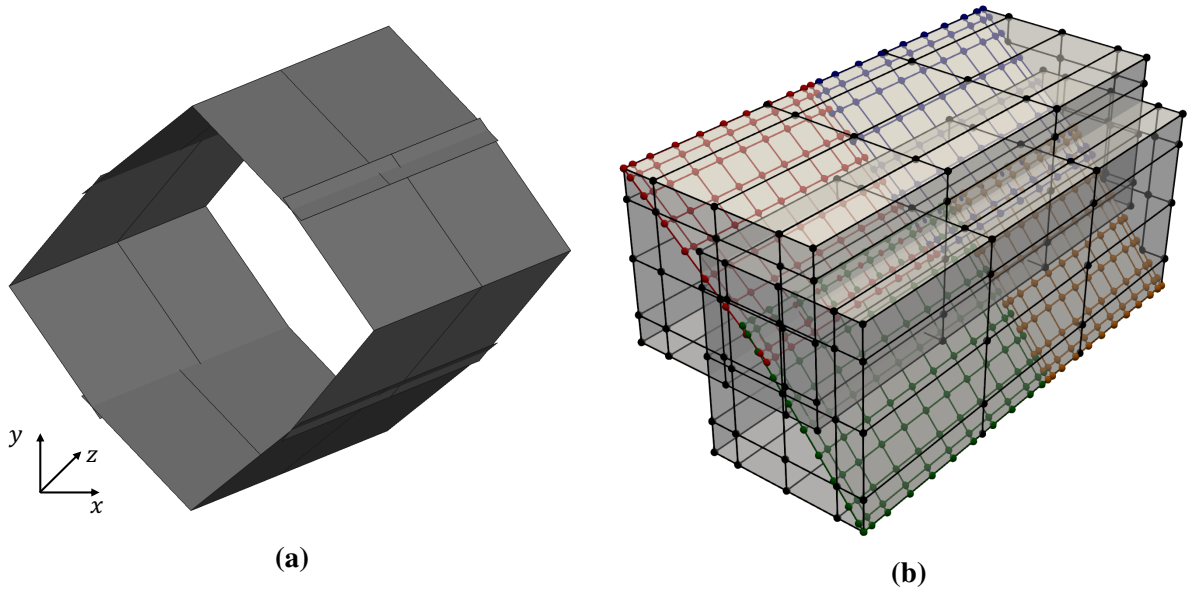


Figure 7.9. (a) The baseline design of a tube geometry, with a quarter of the tube modeled by four non-matching B-spline patches. (b) Two FFD blocks are employed, one for each pair of B-spline surfaces with an edge intersection. Relative movement is allowed between the two FFD blocks.

```
nm_opt.set_shopt_surf_inds_multiFFD(opt_field, opt_surf_inds)
nm_opt.set_geom_preprocessor(preproc)
```

We then create two lists containing the knot vectors `shopt_ffd_knots_list` and control points `shopt_ffd_knots_list` for the trivariate B-spline solids used to define the FFD blocks, as demonstrated in Figure 7.9b. The definitions of these lists are omitted for clarity. The method `set_shopt_multiFFD` is used to obtain the data of the FFD blocks.

```
nm_opt.set_shopt_multiFFD(shopt_ffd_knots_list,
                          shopt_ffd_control_list)
```

Next, the control points of both FFD blocks are aligned in the axial direction using the method `set_shopt_align_CP_multiFFD`. The control points on the faces of the FFD blocks that lie on the symmetric planes are fixed using `set_shopt_pin_CP_multiFFD`. Additionally, the method `set_shopt_regu_CP_multiFFD` is employed to prevent self-penetration of the FFD blocks.

```
nm_opt.set_shopt_align_CP_multiFFD(ffd_ind=0,
                                   align_dir=[[2],[2]])
```

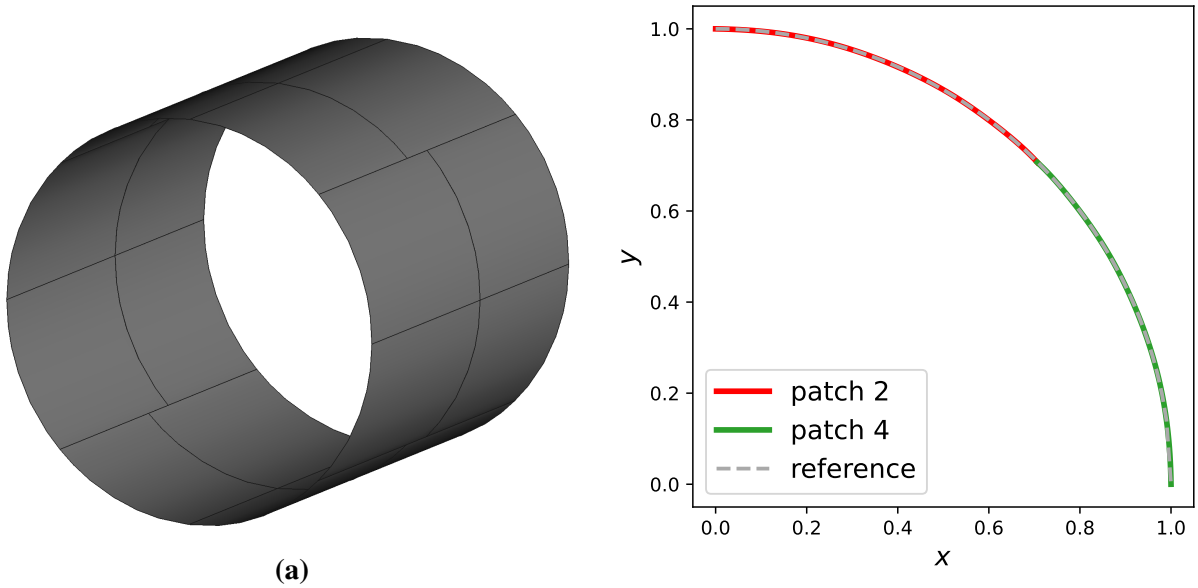
```
nm_opt.set_shopt_align_CP_multiFFD(ffd_ind=1,
                                   align_dir=[[2],[2]])
nm_opt.set_shopt_pin_CP_multiFFD(ffd_ind=0, pin_dir0=[0,0],
                                 pin_side0=[[0],[0]])
nm_opt.set_shopt_pin_CP_multiFFD(ffd_ind=1, pin_dir0=[1,1],
                                 pin_side0=[[0],[0]])
nm_opt.set_shopt_regu_CP_multiFFD()
```

The remainder of the optimization setup is identical to the previous examples, which involves creating quadrature meshes for the intersections and defining PDE residuals for the Kirchhoff–Love shells. The OpenMDAO optimization problem is then created using `om.Problem`. The SNOPT optimizer is employed for this problem with a tolerance of 10^{-2}. The converged tube geometry after 142 iterations is shown in Figure 7.10a. A cross-sectional view of a quarter of the tube is displayed in Figure 7.10b and compared with an exact quarter circle for validation. The surface intersections between the two pairs of shell patches move to the edges of the patches, forming a cylindrical tube to achieve the optimal shape. This demonstrates that both the FFD-based and moving intersections approaches can work simultaneously in shape optimization for non-matching shell structures.

Acknowledgements

A portion of this chapter has been published in "H. Zhao, X. Liu, A. H. Fletcher, R. Xiang, J. T. Hwang, and D. Kamensky. An open-source framework for coupling non-matching isogeometric shells with application to aerospace structures *Computers & Mathematics with Applications*, 111:109–123, 2022." The dissertation author is the primary investigator and author of this paper.

A portion of this chapter is currently being prepared for submission for publication in "H. Zhao, J. T. Hwang, and J. S. Chen. Open-source shape optimization for isogeometric shells using FEniCS and OpenMDAO". The dissertation author was the primary investigator of this



(a) **(b)**

Figure 7.10. (a) The resulting geometry of the tube with minimum internal energy. Surface intersections between the spline patches transit to edge intersections in the optimized design. (b) Comparison of the optimized geometry with an exact cylindrical tube in the cross-sectional view.

material.

Chapter 8

Conclusions and future work

8.1 Conclusions

In this dissertation, a new structural analysis algorithm is developed to perform IGA on shell structures with non-matching parameterizations. Geometrically 2D, topologically 1D quadrature meshes are generated in the parametric space to represent patch intersections and serve as the integration domain of the penalty energy for imposition of interaction kinematic compatibility. The implementation leverages robust computational geometry operations from the OpenCASCADE geometry kernel and modern code generation capabilities from the FEniCS Project. The use of code generation is especially useful both for future extensions of the analysis framework and for applications in design optimization.

In the context of design optimization, the use of FEniCS's code generation provides efficient and automatic access to derivatives of the Kirchhoff–Love shell discrete systems with respect to state and design variables, as needed by efficient minimization algorithms [50]. Novel shape optimization approaches are proposed for shell structures composed of multiple untrimmed B-spline or NURBS patches. The FFD-based shape and thickness optimization approach is developed for shell structures composed of separately parametrized NURBS surfaces. The integration of this method with the Lagrange extraction technique enables IGA with existing FE toolkits and provides a connection between the FFD block and non-matching shell patches. By employing the FFD block approach, the updated shell geometry and thickness compatibility

are properly maintained at patch intersections throughout the optimization process. This feature prevents undesired shape discontinuities in shell structural optimization. The automation of analytical derivative computations is achieved through code generation in FEniCS, enabling gradient-based multidisciplinary design optimization. This automation streamlines the optimization process and allows for efficient exploration of design spaces. The unified NURBS representation shared by both the design geometry and analysis model enhances accuracy per DoF in the analysis and precise design updates. Moreover, the proposed framework circumvents FE mesh generation and streamlines design-analysis-optimization workflow for complex shell structures. Consequently, the automated workflow accelerates the conceptual design of novel eVTOL aircraft with minimal manual effort. A suite of benchmark problems is adopted to verify the effectiveness of the FFD-based optimization approach proposed in this dissertation. Both the shape optimization and thickness optimization results agree well with analytical solutions or other established references. Furthermore, we have applied the framework to two different aircraft wings. This demonstration highlights the potential of the proposed method in exploring complex design spaces and obtaining superior designs for innovative aircraft structures.

The moving intersection shape optimization approach allows relative movement between shell patches without compromising mesh quality during shape updates, thereby enabling moving intersections. To achieve the moving intersections during design optimization, partial derivatives of the penalty energy with respect to surface intersections' parametric locations are formulated, along with an implicit relation between shell patches' control points and surface intersections' parametric locations. Standard benchmark problems are employed to validate this shape optimization approach, demonstrating that the optimized solutions closely match the reference solutions. Furthermore, we apply the proposed approach to adjust the layout of the internal spars and ribs of an eVTOL aircraft wing, resulting in nontraditional wing designs aimed at minimizing the internal energy of the wing.

Integrating IGA into shell shape optimization presents notable advantages. The direct analysis based on CAD geometries in IGA naturally bridges the gap between the design model

and analysis model within the optimization loop without geometric errors. The coarse design model is employed to update the shape of the geometry, while the fine analysis model is used for structural analysis. Shape modifications are directly applied to the coarse design model, and the structural response of the updated geometry is evaluated using the refined analysis model. As such, the dimension of the design space can be significantly reduced. This shape optimization workflow for non-matching shells is significantly simplified and accelerates the conceptual design of complex shell structures. The optimal designs provide valuable insights for the development of innovative shell structures.

Open-source Python libraries are developed for direct structural analysis and design optimization of complex shell structures. The structural analysis employs an isogeometric Kirchhoff–Love shell model coupled with a penalty formulation for patch intersections, and control points of shell patches are modified to update the shell shape. In this approach, FE mesh generation is no longer required in the optimization loop. This framework is developed based on FEniCS, leveraging its automatic differentiation and code generation capabilities to compute discrete analytical derivatives, thereby enabling gradient-based design optimization. The code framework accepts B-spline or NURBS-based CAD geometries as input and returns the optimized geometry, streamlining the workflow from geometry design through structural analysis to shape optimization. The modular design of the optimization framework inherited from OpenMDAO is presented along with the essential components for FFD-based shape optimization and the moving intersections approaches. A suite of benchmark problems, accompanied by code implementations, is provided to examine the effectiveness of the framework.

8.2 Future work

The recommendations for future work are summarized as follows.

- Development of more physical design optimization coupling with other solvers, such as aerodynamics [165, 164, 143], electric motors [151], and noise control [59]. This presents

opportunities for future work on integrating various disciplines to enhance the framework for more practical shell structure designs.

- Development of coupling methods for different types of isogeometric structures, including shell–beam coupling and shell–solid coupling, for applications to aerospace structure design, such as spar caps and stringers.
- Development of analysis and optimization frameworks coupling with reproducing kernel particle method (RKPM) to leverage adaptive refinements, thereby predicting the stress concentrations accurately during analysis and optimization iterations.
- Development of shape optimization methods for handling moving intersections with changing topology, including allowing intersections to cross patch boundaries and the ability to add or remove intersections during the shape optimization process.
- Development of functionality to incorporate additional types of splines in IGA, including trimmed NURBS surfaces and T-spline surfaces, to better accommodate more complex CAD geometries.
- Investigation of reduced-order modeling (ROM) methods for complex isogeometric shells with patch coupling to enhance computational efficiency.

Acknowledgements

A portion of this chapter has been published in "H. Zhao, X. Liu, A. H. Fletcher, R. Xiang, J. T. Hwang, and D. Kamensky. An open-source framework for coupling non-matching isogeometric shells with application to aerospace structures *Computers & Mathematics with Applications*, 111:109–123, 2022." The dissertation author is the primary investigator and author of this paper.

A portion of this chapter has been published in "H. Zhao, D. Kamensky, J. T. Hwang, and J. S. Chen. Automated shape and thickness optimization for non-matching isogeometric

shells using free-form deformation. *Engineering with Computers*, 1-24, 2024." The dissertation author is the primary investigator and author of this paper.

A portion of this chapter has been published in "H. Zhao, J. T. Hwang, and J. S. Chen. Shape optimization of non-matching isogeometric shells with moving intersections. *Computer Methods in Applied Mechanics and Engineering*, 431:117322, 2024." The dissertation author is the primary investigator and author of this paper.

A portion of this chapter is currently being prepared for submission for publication in "H. Zhao, J. T. Hwang, and J. S. Chen. Open-source shape optimization for isogeometric shells using FEniCS and OpenMDAO". The dissertation author was the primary investigator of this material.

Bibliography

[1] *COMSOL Multiphysics Reference Manual, version 5.6.* COMSOL, Inc.

[2] E. J. Adler, B. J. Brelje, and J. R. R. A. Martins. Thermal management system optimization for a parallel hybrid aircraft considering mission fuel burn. *Aerospace*, 9(5), April 2022.

[3] M. S. Alnæs, A. Logg, K. B. Ølgaard, M. E. Rognes, and G. N. Wells. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Trans. Math. Softw.*, 40(2):9:1–9:37, March 2014.

[4] H. Azegami, S. Fukumoto, and T. Aoyama. Shape optimization of continua using NURBS as basis functions. *Structural and Multidisciplinary Optimization*, 47:247–258, 2013.

[5] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.6, Argonne National Laboratory, 2015.

[6] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, and H. Zhang. PETSc Web page. http://www.mcs.anl.gov/petsc, 2015.

[7] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[8] K.-J. Bathe and E. N. Dvorkin. A formulation of general shell elements—the use of mixed interpolation of tensorial components. *International journal for numerical methods in engineering*, 22(3):697–722, 1986.

[9] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, T. J. R. Hughes, S. Lipton, M. A. Scott, and T. W. Sederberg. Isogeometric analysis using T-splines. *Computer methods in applied mechanics and engineering*, 199(5-8):229–263, 2010.

[10] Y. Bazilevs, M.-C. Hsu, J. Kiendl, R. Wüchner, and K.-U. Bletzinger. 3D simulation of wind turbine rotors at full scale. part II: Fluid–structure interaction modeling with composite blades. *International Journal for Numerical Methods in Fluids*, 65(1-3):236–253, 2011.

[11] Y. Bazilevs, D. Kamensky, G. Moutsanidis, and S. Shende. Residual-based shock capturing in solids. *Computer Methods in Applied Mechanics and Engineering*, 358:112638, 2020.

[12] Y. Bazilevs, K. Takizawa, T. E. Tezduyar, M.-C. Hsu, Y. Otoguro, H. Mochizuki, and M. C. H. Wu. Wind turbine and turbomachinery computational analysis with the ale and space-time variational multiscale methods and isogeometric discretization. *Journal of Advanced Engineering and Computation*, 4(1):1–32, 2020.

[13] T. Belytschko and M. Hillman. *Meshfree and particle methods: fundamentals and applications*. John Wiley & Sons, 2023.

[14] T. Belytschko, W. K. Liu, B. Moran, and K. Elkhodary. *Nonlinear finite elements for continua and structures*. John wiley & sons, 2014.

[15] T. Belytschko, Y. Y. Lu, and L. Gu. Element-free Galerkin methods. *International journal for numerical methods in engineering*, 37(2):229–256, 1994.

[16] T. Belytschko, H. Stolarski, W. K. Liu, N. Carpenter, and J. S.-J. Ong. Stress projection for membrane and shear locking in shell finite elements. *Computer Methods in Applied Mechanics and Engineering*, 51:221–258, 1985.

[17] M. P. Bendsoe and O. Sigmund. *Topology optimization: theory, methods, and applications*. Springer Science & Business Media, 2013.

[18] J. Benzaken, J. A. Evans, S. F. McCormick, and R. Tamstorf. Nitsche's method for linear Kirchhoff–Love shells: Formulation, error analysis, and verification. *Computer Methods in Applied Mechanics and Engineering*, 374:113544, 2021.

[19] S. Bieber, B. Oesterle, E. Ramm, and M. Bischoff. A variational method to avoid locking—independent of the discretization scheme. *International Journal for Numerical Methods in Engineering*, 114(8):801–827, 2018.

[20] K.-U. Bletzinger and E. Ramm. Form finding of shells by structural optimization. *Engineering with computers*, 9:27–35, 1993.

[21] K.-U. Bletzinger, R. Wüchner, F. Daoud, and N. Camprubí. Computational methods for form finding and optimization of shells and membranes. *Computer methods in applied mechanics and engineering*, 194(30-33):3438–3452, 2005.

[22] J. Bleyer. *Numerical Tours of Computational Mechanics with FEniCS*, 2018.

[23] M. J. Borden, M. A. Scott, J. A. Evans, and T. J. R. Hughes. Isogeometric finite element data structures based on Bézier extraction of NURBS. *International Journal for Numerical Methods in Engineering*, 87:15–47, 2011.

[24] R. Bouclier and T. Hirschler. *IGA: Non-conforming coupling and shape optimization of complex multipatch structures*. John Wiley & Sons, 2022.

[25] E. Brivadis, A. Buffa, B. Wohlmuth, and L. Wunderlich. Isogeometric mortar methods. *Computer Methods in Applied Mechanics and Engineering*, 284:292–319, 2015. Isogeometric Analysis Special Issue.

[26] E. M. B. Campello, P. M. Pimenta, and P. Wriggers. A triangular finite shell element based on a fully nonlinear shell formulation. *Computational Mechanics*, 31(6):505–518, Aug 2003.

[27] H. Casquero and M. Golestanian. Removing membrane locking in quadratic NURBS-based discretizations of linear plane Kirchhoff rods: CAS elements. *Computer Methods in Applied Mechanics and Engineering*, 399:115354, 2022.

[28] H. Casquero, L. Liu, Y. Zhang, A. Reali, J. Kiendl, and H. Gomez. Arbitrary-degree T-splines for isogeometric analysis of fully nonlinear Kirchhoff–Love shells. *Computer-Aided Design*, 82:140–153, 2017. Isogeometric Design and Analysis.

[29] H. Casquero and K. D. Mathews. Overcoming membrane locking in quadratic NURBS-based discretizations of linear Kirchhoff–Love shells: CAS elements. *Computer Methods in Applied Mechanics and Engineering*, 417:116523, 2023.

[30] H. Casquero, X. Wei, D. Toshniwal, A. Li, T. J. R. Hughes, J. Kiendl, and Y. J. Zhang. Seamless integration of design and Kirchhoff–Love shell analysis using analysis-suitable unstructured T-splines. *Computer Methods in Applied Mechanics and Engineering*, 360:112765, 2020.

[31] M. Chasapi, P. Antolin, and A. Buffa. Fast parametric analysis of trimmed multi-patch isogeometric Kirchhoff-Love shells using a local reduced basis method. *Engineering with Computers*, pages 1–28, 2024.

[32] J.-S. Chen, M. Hillman, and S.-W. Chi. Meshfree methods: progress made after 20 years. *Journal of Engineering Mechanics*, 143(4):04017001, 2017.

[33] J.-S. Chen, M. Hillman, and M. Rüter. An arbitrary order variationally consistent integration for Galerkin meshfree methods. *International Journal for Numerical Methods in Engineering*, 95(5):387–418, 2013.

[34] J.-S. Chen and N. H. Kim. Meshfree method and application to shape optimization. In *Optimization of Structural and Mechanical Systems*, pages 389–414. World Scientific, 2007.

[35] J.-S. Chen, C. Pan, C. Roque, and H.-P. Wang. A lagrangian reproducing kernel particle method for metal forming analysis. *Computational mechanics*, 22:289–307, 1998.

[36] J.-S. Chen, C. Pan, and C.-T. Wu. Large deformation analysis of rubber based on a reproducing kernel particle method. *Computational Mechanics*, 19(3):211–227, 1997.

[37] J.-S. Chen, C. Pan, C.-T. Wu, and W. K. Liu. Reproducing kernel particle methods for large deformation analysis of non-linear structures. *Computer methods in applied mechanics and engineering*, 139(1-4):195–227, 1996.

[38] J.-S. Chen, C.-T. Wu, S. Yoon, and Y. You. A stabilized conforming nodal integration for Galerkin mesh-free methods. *International journal for numerical methods in engineering*, 50(2):435–466, 2001.

[39] J.-S. Chen, S. Yoon, and C.-T. Wu. Non-linear version of stabilized conforming nodal integration for Galerkin mesh-free methods. *International Journal for Numerical Methods in Engineering*, 53(12):2587–2615, 2002.

[40] Y. Chen, S. Jun, J.-S. Chen, T. Belytschko, C. Pan, and R. A. Uras. Overview and applications of the reproducing kernel particle methods. *Archives of Computational Methods in Engineering*, 3:3–80, 1996.

[41] S. Cho and S.-H. Ha. Isogeometric shape design optimization: exact geometry and enhanced sensitivity. *Structural and Multidisciplinary Optimization*, 38:53–70, 2009.

[42] H. Chung, J. T. Hwang, J. S. Gray, and H. A. Kim. Topology optimization in OpenMDAO. *Structural and multidisciplinary optimization*, 59:1385–1400, 2019.

[43] L. Coradello, J. Kiendl, and A. Buffa. Coupling of non-conforming trimmed isogeometric Kirchhoff–Love shells via a projected super-penalty approach, 2021.

[44] L. Coradello, G. Loli, and A. Buffa. A projected super-penalty method for the C^1-coupling of multi-patch isogeometric Kirchhoff plates. *Computational Mechanics*, 67(4):1133–1153, Apr 2021.

[45] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. Wiley, Chichester, 2009.

[46] Y. Ding. Shape optimization of structures: a literature survey. *Computers & Structures*, 24(6):985–1004, 1986.

[47] T. X. Duong, F. Roohbakhshan, and R. A. Sauer. A new rotation-free isogeometric thin shell formulation and a corresponding continuity constraint for patch boundaries. *Computer Methods in Applied Mechanics and Engineering*, 316:43–83, 2017. Special Issue on Isogeometric Analysis: Progress and Challenges.

[48] H. A. Eschenauer and N. Olhoff. Topology optimization of continuum structures: a review. *Appl. Mech. Rev.*, 54(4):331–390, 2001.

[49] J. A. Evans, Y. Bazilevs, I. Babuška, and T. J. R. Hughes. n-widths, sup–infs, and optimality ratios for the k-version of the isogeometric finite element method. *Computer Methods in Applied Mechanics and Engineering*, 198(21):1726–1741, 2009. Advances in Simulation-Based Engineering Sciences – Honoring J. Tinsley Oden.

[50] P. E. Farrell, D. A. Ham, S. W. Funke, and M. E. Rognes. Automated derivation of the adjoint of high-level transient finite element programs. *SIAM Journal on Scientific Computing*, 35(4):C369–C393, 2013.

[51] M. Farshad. *Design and analysis of shell structures*, volume 16. Springer Science & Business Media, Switzerland, 2013.

[52] W. Fredericks, K. Antcliff, G. Costa, N. Deshpande, M. Moore, E. San Miguel, and A. Snyder. Aircraft conceptual design using vehicle sketch pad. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2010.

[53] J. E. Fromm, N. Wunsch, K. Maute, J. A. Evans, and J.-S. Chen. Interpolation-based immersogeometric analysis methods for multi-material and multi-physics problems. *Computational Mechanics*, pages 1–25, 2024.

[54] J. E. Fromm, N. Wunsch, R. Xiang, H. Zhao, K. Maute, J. A. Evans, and D. Kamensky. Interpolation-based immersed finite element and isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 405:115890, 2023.

[55] V. Gandarillas, A. J. Joshy, M. Z. Sperry, A. K. Ivanov, and J. T. Hwang. A graph-based methodology for constructing computational models that automates adjoint-based sensitivity analysis. *Structural and Multidisciplinary Optimization*, 67(5):76, 2024.

[56] R. Ghaffari, T. X. Duong, and R. A. Sauer. A new shell formulation for graphene structures based on existing ab-initio data. *International Journal of Solids and Structures*, 135:37–60, 2018.

[57] R. Ghaffari and R. A. Sauer. A new efficient hyperelastic finite element model for graphene and its application to carbon nanotubes and nanocones. *Finite Elements in Analysis and Design*, 146:42–61, 2018.

[58] C. Giannelli, B. Jüttler, and H. Speleers. THB-splines: The truncated basis for hierarchical splines. *Computer Aided Geometric Design*, 29(7):485–498, 2012.

[59] H. Gill, S. Lee, M. L. Ruh, and J. T. Hwang. Applicability of low-fidelity tonal and broadband noise models on small-scaled rotors. In *AIAA SciTech 2023 Forum*, page 1547, 2023.

[60] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.

[61] J. Gloudemans, P. Davis, and P. Gelhausen. A rapid geometry modeler for conceptual aircraft. In *34th Aerospace Sciences Meeting and Exhibit*, 1996.

[62] J. Gloudemans and R. McDonald. Improved geometry modeling for high fidelity parametric design. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2010.

[63] J. S. Gray, J. T. Hwang, J. R. R. A. Martins, K. T. Moore, and B. A. Naylor. OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization. *Structural and Multidisciplinary Optimization*, 59:1075–1104, 2019.

[64] L. Greco, M. Cuomo, and L. Contrafatto. A reconstructed local B formulation for isogeometric Kirchhoff–Love shells. *Computer Methods in Applied Mechanics and Engineering*, 332:462–487, 2018.

[65] L. Greco, M. Cuomo, L. Contrafatto, and S. Gazzo. An efficient blended mixed B-spline formulation for removing membrane locking in plane curved Kirchhoff rods. *Computer Methods in Applied Mechanics and Engineering*, 324:476–511, 2017.

[66] I. Grindeanu, N. H. Kim, K. K. Choi, and J.-S. Chen. CAD-based shape optimization using a meshfree method. *Concurrent Engineering*, 10(1):55–66, 2002.

[67] G. Guarino, P. Antolin, A. Milazzo, and A. Buffa. An interior penalty coupling strategy for isogeometric non-conformal Kirchhoff–Love shell patches. *Engineering With Computers*, pages 1–27, 2024.

[68] Y. Guo, J. Heller, T. J. R. Hughes, M. Ruess, and D. Schillinger. Variationally consistent isogeometric analysis of trimmed thin shells at finite deformations, based on the STEP exchange format. *Computer Methods in Applied Mechanics and Engineering*, 336:39–79, 2018.

[69] Y. Guo and M. Ruess. Nitsche's method for a coupling of isogeometric thin shells and blended shell structures. *Computer Methods in Applied Mechanics and Engineering*, 284:881–905, 2015.

[70] Y. Guo, Z. Zou, and M. Ruess. Isogeometric multi-patch analyses for mixed thin shells in the framework of non-linear elasticity. *Computer Methods in Applied Mechanics and Engineering*, 380:113771, 2021.

[71] H. Gómez, V. M. Calo, Y. Bazilevs, and T. J. R. Hughes. Isogeometric analysis of the Cahn–Hilliard phase-field model. *Computer Methods in Applied Mechanics and Engineering*, 197(49):4333–4352, 2008.

[72] S.-H. Ha, K. K. Choi, and S. Cho. Numerical method for shape optimization using T-spline based isogeometric method. *Structural and Multidisciplinary Optimization*, 42:417–428, 2010.

[73] A. Hahn. Vehicle sketch pad: A parametric geometry modeler for conceptual aircraft design. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, 2010.

[74] P. Hao, Y. Wang, L. Jin, S. Ma, and B. Wang. An isogeometric design-analysis-optimization workflow of stiffened thin-walled structures via multilevel NURBS-based free-form deformations (MNFFD). *Computer Methods in Applied Mechanics and Engineering*, 408:115936, 2023.

[75] P. Hao, X. Yuan, C. Liu, B. Wang, H. Liu, G. Li, and F. Niu. An integrated framework of exact modeling, isogeometric analysis and optimization for variable-stiffness composite panels. *Computer Methods in Applied Mechanics and Engineering*, 339:205–238, 2018.

[76] M. F. Hardwick, R. L. Clay, P. T. Boggs, E. J. Walsh, A. R. Larzelere, and A. Altshuler. DART system analysis. Technical Report SAND2005-4647, Sandia National Laboratories, 2005.

[77] A. J. Herrema, E. L. Johnson, D. Proserpio, M. C. H. Wu, J. Kiendl, and M.-C. Hsu. Penalty coupling of non-matching isogeometric Kirchhoff–Love shell patches with application to composite wind turbine blades. *Computer Methods in Applied Mechanics and Engineering*, 346:810–840, 2019.

[78] A. J. Herrema, J. Kiendl, and M.-C. Hsu. A framework for isogeometric-analysis-based optimization of wind turbine blade structures. *Wind Energy*, 22(2):153–170, 2019.

[79] A. J. Herrema, N. M. Wiese, C. N. Darling, B. Ganapathysubramanian, A. Krishnamurthy, and M.-C. Hsu. A framework for parametric design optimization using isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 316:944–965, 2017.

[80] M. Hillman and J.-S. Chen. An accelerated, convergent, and stable nodal integration in Galerkin meshfree methods for linear and nonlinear mechanics. *International Journal for Numerical Methods in Engineering*, 107(7):603–630, 2016.

[81] T. Hirschler. *Isogeometric modeling for the optimal design of aerostructures*. PhD thesis, Université de Lyon, 2019.

[82] T. Hirschler, R. Bouclier, D. Dureisseix, A. Duval, T. Elguedj, and J. Morlier. A dual domain decomposition algorithm for the analysis of non-conforming isogeometric Kirchhoff–Love shells. *Computer Methods in Applied Mechanics and Engineering*, 357:112578, 2019.

[83] T. Hirschler, R. Bouclier, A. Duval, T. Elguedj, and J. Morlier. The embedded isogeometric Kirchhoff–Love shell: From design to shape optimization of non-conforming stiffened multipatch structures. *Computer Methods in Applied Mechanics and Engineering*, 349:774–797, 2019.

[84] T. Hirschler, R. Bouclier, A. Duval, T. Elguedj, and J. Morlier. A new lighting on analytical discrete sensitivities in the context of isogeometric shape optimization. *Archives of Computational Methods in Engineering*, 28(4):2371–2408, 2021.

[85] T. Horger, A. Reali, B. Wohlmuth, and L. Wunderlich. A hybrid isogeometric approach on multi-patches with applications to Kirchhoff plates and eigenvalue problems. *Computer Methods in Applied Mechanics and Engineering*, 348:396–408, 2019.

[86] M.-C. Hsu, C. Wang, F. Xu, A. J. Herrema, and A. Krishnamurthy. Direct immersogeometric fluid flow analysis using B-rep CAD models. *Computer Aided Geometric Design*, 43:143–158, 2016. Geometric Modeling and Processing 2016.

[87] T. J. R. Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

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

[89] J. T. Hwang and J. R. R. A. Martins. GeoMACH: Geometry-centric MDAO of aircraft configurations with high fidelity. In *12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2012.

[90] M. H. Imam. Three-dimensional shape optimization. *International Journal for Numerical Methods in Engineering*, 18(5):661–673, 1982.

[91] J. P. Jasa, B. J. Brelje, J. S. Gray, C. A. Mader, and J. R. R. A. Martins. Large-scale path-dependent optimization of supersonic aircraft. *Aerospace*, 7(10):152, 2020.

[92] J. P. Jasa, J. T. Hwang, and J. R. R. A. Martins. Open-source coupled aerostructural optimization using Python. *Structural and Multidisciplinary Optimization*, 57(4):1815–1827, April 2018.

[93] C. M. Jauregui, S. L. Schnulo, J. S. Gray, and H. A. Kim. Level set topology optimization in OpenMDAO. In *AIAA SciTech 2023 Forum*, 2023.

[94] E. L. Johnson and M.-C. Hsu. Isogeometric analysis of ice accretion on wind turbine blades. *Computational Mechanics*, 66(2):311–322, Aug 2020.

[95] E. L. Johnson, D. W. Laurence, F. Xu, C. E. Crisp, A. Mir, H. M. Burkhart, C.-H. Lee, and M.-C. Hsu. Parameterization, geometric modeling, and isogeometric analysis of tricuspid valves. *Computer Methods in Applied Mechanics and Engineering*, 384:113960, 2021.

[96] I. Kaljević and S. Saigal. An improved element free Galerkin formulation. *International Journal for numerical methods in engineering*, 40(16):2953–2974, 1997.

[97] D. Kamensky. Open-source immersogeometric analysis of fluid–structure interaction using FEniCS and tIGAr. *Computers & Mathematics with Applications*, 81:634–648, 2021. Development and Application of Open-source Software for Problems with Numerical PDEs.

[98] D. Kamensky and Y. Bazilevs. tIGAr: Automating isogeometric analysis with FEniCS. *Computer Methods in Applied Mechanics and Engineering*, 344:477–498, 2019.

[99] D. Kamensky, J. A. Evans, M.-C. Hsu, and Y. Bazilevs. Projection-based stabilization of interface Lagrange multipliers in immersogeometric fluid–thin structure interaction analysis, with application to heart valve modeling. *Computers & Mathematics with Applications*, 74(9):2068–2088, 2017. Advances in Mathematics of Finite Elements, honoring 90th birthday of Ivo Babuška.

[100] D. Kamensky, M.-C. Hsu, D. Schillinger, J. A. Evans, A. Aggarwal, Y. Bazilevs, M. S. Sacks, and T. J. R. Hughes. An immersogeometric variational framework for fluid–structure interaction: Application to bioprosthetic heart valves. *Computer Methods in Applied Mechanics and Engineering*, 284:1005–1053, 2015.

[101] D. Kamensky, F. Xu, C.-H. Lee, J. Yan, Y. Bazilevs, and M.-C. Hsu. A contact formulation based on a volumetric potential: Application to isogeometric simulations of atrioventricular valves. *Computer Methods in Applied Mechanics and Engineering*, 330:522–546, 2018.

[102] G. J. Kennedy and J. E. Hicken. Improved constraint-aggregation methods. *Computer Methods in Applied Mechanics and Engineering*, 289:332–354, 2015.

[103] J. Kiendl. *Isogeometric Analysis and Shape Optimal Design of Shell Structures*. PhD thesis, Lehrstuhl für Statik, Technische Universität München, 2011.

[104] J. Kiendl, Y. Bazilevs, M.-C. Hsu, R. Wüchner, and K.-U. Bletzinger. The bending strip method for isogeometric analysis of Kirchhoff–Love shell structures comprised of multiple patches. *Computer Methods in Applied Mechanics and Engineering*, 199:2403–2416, 2010.

[105] J. Kiendl, K.-U. Bletzinger, J. Linhard, and R. Wüchner. Isogeometric shell analysis with Kirchhoff–Love elements. *Computer Methods in Applied Mechanics and Engineering*, 198(49-52):3902–3914, 2009.

[106] J. Kiendl, M.-C. Hsu, M. C. H. Wu, and A. Reali. Isogeometric Kirchhoff–Love shell formulations for general hyperelastic materials. *Computer Methods in Applied Mechanics and Engineering*, 291(0):280–303, 2015.

[107] J. Kiendl, R. Schmidt, R. Wüchner, and K.-U. Bletzinger. Isogeometric shape optimization of shells using semi-analytical sensitivity analysis and sensitivity weighting. *Computer Methods in Applied Mechanics and Engineering*, 274:148–167, 2014.

[108] N. H. Kim, K. K. Choi, and J.-S. Chen. Shape design sensitivity analysis and optimization of elasto-plasticity with frictional contact. *AIAA journal*, 38(9):1742–1753, 2000.

[109] N. H. Kim, K. K. Choi, and J.-S. Chen. Die shape design optimization of sheet metal stamping process using meshfree method. *International Journal for Numerical Methods in Engineering*, 51(12):1385–1405, 2001.

[110] N. H. Kim, K. K. Choi, and J.-S. Chen. Structural optimization of finite deformation elastoplasticity using continuum-based shape design sensitivity formulation. *Computers & Structures*, 79(20-21):1959–1976, 2001.

[111] N. H. Kim, K. K. Choi, J.-S. Chen, and Y. H. Park. Meshless shape design sensitivity analysis and optimization for contact problem with friction. *Computational Mechanics*, 25:157–168, 2000.

[112] R. C. Kirby and A. Logg. A compiler for variational forms. *ACM Trans. Math. Softw.*, 32(3):417–444, September 2006.

[113] D. Kraft. A software package for sequential quadratic programming. *Forschungsbericht-Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt*, 1988.

[114] P. Krysl and T. Belytschko. Analysis of thin plates by the element-free Galerkin method. *Computational Mechanics*, 17(1):26–35, 1995.

[115] P. Krysl and T. Belytschko. Analysis of thin shells by the element-free Galerkin method. *International Journal of Solids and Structures*, 33(20-22):3057–3080, 1996.

[116] L. Leonetti, F. S. Liguori, D. Magisano, J. Kiendl, A. Reali, and G. Garcea. A robust penalty coupling of non-matching isogeometric Kirchhoff–Love shell patches in large deformations. *Computer Methods in Applied Mechanics and Engineering*, 371:113289, 2020.

[117] K. Li and X. Qian. Isogeometric analysis and shape optimization via boundary integral. *Computer-Aided Design*, 43(11):1427–1437, 2011.

[118] J.-T. Lin, C. Girerd, J. Yan, J. T. Hwang, and T. K. Morimoto. A generalized framework for concentric tube robot design using gradient-based optimization. *IEEE Transactions on Robotics*, 38(6):3774–3791, 2022.

[119] W. Liu, Z. Deng, J. Li, and X. Hu. Investigating the electrothermal behavior of evtol batteries in urban air mobility applications. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 40–45, 2022.

[120] W. K. Liu, S. Jun, S. Li, J. Adee, and T. Belytschko. Reproducing kernel particle methods for structural dynamics. *International Journal for Numerical Methods in Engineering*, 38(10):1655–1679, 1995.

[121] W. K. Liu, S. Jun, and Y. F. Zhang. Reproducing kernel particle methods. *International journal for numerical methods in fluids*, 20(8-9):1081–1106, 1995.

[122] A. Logg, K.-A. Mardal, and G. Wells. *Automated solution of differential equations by the finite element method: The FEniCS book*, volume 84. Springer Science & Business Media, 2012.

[123] A. Logg and G. N. Wells. DOLFIN: Automated finite element computing. *ACM Trans. Math. Softw.*, 37(2):20:1–20:28, April 2010.

[124] Y. Y. Lu, T. Belytschko, and L. Gu. A new implementation of the element free Galerkin method. *Computer methods in applied mechanics and engineering*, 113(3-4):397–414, 1994.

[125] R. H. Macneal and R. L. Harder. A proposed standard set of problems to test finite element accuracy. *Finite Elements in Analysis and Design*, 1(1):3–20, 1985.

[126] D. Magisano, A. Corrado, L. Leonetti, J. Kiendl, and G. Garcea. Large deformation Kirchhoff–Love shell hierarchically enriched with warping: Isogeometric formulation and modeling of alternating stiff/soft layups. *Computer Methods in Applied Mechanics and Engineering*, 418:116556, 2024.

[127] S. Morganti, F. Auricchio, D. J. Benson, F. I. Gambarin, S. Hartmann, T. J. R. Hughes, and A. Reali. Patient-specific isogeometric structural analysis of aortic valve closure. *Computer Methods in Applied Mechanics and Engineering*, 284:508–520, 2015.

[128] A. P. Nagy, M. M. Abdalla, and Z. Gürdal. Isogeometric sizing and shape optimisation of beam structures. *Computer Methods in Applied Mechanics and Engineering*, 199(17-20):1216–1230, 2010.

[129] A. P. Nagy, S. T. IJsselmuiden, and M. M. Abdalla. Isogeometric design of anisotropic shells: optimal form and material distribution. *Computer Methods in Applied Mechanics and Engineering*, 264:145–162, 2013.

[130] G. E. Neighbor, H. Zhao, M. Saraeian, M.-C. Hsu, and D. Kamensky. Leveraging code generation for transparent immersogeometric fluid–structure interaction analysis on deforming domains. *Engineering with Computers*, 39(2):1019–1040, 2023.

[131] A. Neofytou, T.-H. Huang, S. Kambampati, R. Picelli, J.-S. Chen, and H. A. Kim. Level set topology optimization with nodally integrated reproducing kernel particle method. *Computer Methods in Applied Mechanics and Engineering*, 385:114016, 2021.

[132] A. Neofytou, R. Picelli, J.-S. Chen, and H. A. Kim. Level set topology optimization for design dependent pressure loads: a comparison between FEM and RKPM. In *AIAA Aviation 2019 Forum*.

[133] A. Neofytou, R. Picelli, T.-H. Huang, J.-S. Chen, and H. A. Kim. Level set topology optimization for design-dependent pressure loads using the reproducing kernel particle method. *Structural and Multidisciplinary Optimization*, 61:1805–1820, 2020.

[134] T.-H. Nguyen, R. R. Hiemstra, and D. Schillinger. Leveraging spectral analysis to elucidate membrane locking and unlocking in isogeometric finite element formulations of the curved Euler–Bernoulli beam. *Computer Methods in Applied Mechanics and Engineering*, 388:114240, 2022.

[135] N. Nguyen-Thanh, J. Kiendl, H. Nguyen-Xuan, R. Wüchner, K.-U. Bletzinger, Y. Bazilevs, and T. Rabczuk. Rotation free isogeometric thin shell analysis using PHT-splines. *Computer Methods in Applied Mechanics and Engineering*, 200(47):3410–3424, 2011.

[136] N. Nguyen-Thanh, K. Zhou, X. Zhuang, P. Areias, H. Nguyen-Xuan, Y. Bazilevs, and T. Rabczuk. Isogeometric analysis of large-deformation thin shells using RHT-splines for multiple-patch coupling. *Computer Methods in Applied Mechanics and Engineering*, 316:1157–1178, 2017.

[137] T. Paviot and J. Feringa. Pythonocc. Technical report, 3D CAD/CAE/PLM development framework for the Python programming language, 2018.

[138] L. Piegl and W. Tiller. *The NURBS book*. Springer Science & Business Media, 2012.

[139] N. Polaczyk, E. Trombino, P. Wei, and M. Mitici. A review of current technology and research in urban on-demand air mobility applications. In *Vertical Flight Society Autonomous VTOL Technical Meeting and Electric VTOL Symposium*, 2019.

[140] D. Proserpio and J. Kiendl. Penalty coupling of trimmed isogeometric Kirchhoff–Love shell patches. *Journal of Mechanics*, 38:156–165, 2022.

[141] M. A. Puso, J.-S. Chen, E. Zywicz, and W. Elmer. Meshfree and finite element nodal integration methods. *International Journal for Numerical Methods in Engineering*, 74(3):416–446, 2008.

[142] X. Qian. Full analytical sensitivities in NURBS based isogeometric shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 199(29-32):2059–2071, 2010.

[143] M. L. Ruh, A. Fletcher, D.Sarojini, M. Sperry, J. Yan, L. Scotzniovsky, S. P. van Schie, M. Warner, N. C. Orndorff, R. Xiang, A. J. Joshy, H. Zhao, J. Krokowski, H. Gill, S. Lee, Z. Cheng, Z. Cao, C. Mi, C. Silva, L. Wolfe, J.-S. Chen, and J. T. Hwang. Large-scale multidisciplinary design optimization of a NASA air taxi concept using a comprehensive physics-based system model. In *AIAA SciTech 2024 Forum*, page 0771, 2024.

[144] D. Sarojini, M. L. Ruh, J. Yan, L. Scotzniovsky, N. C. Orndorff, R. Xiang, H. Zhao, J. Krokowski, M. Warner, S. P. van Schie, A. Cronk, A. T. Guibert, J. T. Chambers, L. Wolfe, R. Doring, R. Despins, C. Joseph, R. Anderson, A. Ning, H. Gill, S. Lee, Z. Cheng, Z. Cao, C. Mi, Y. S. Meng, C. Silva, J.-S. Chen, A. A. Kim, and J. T. Hwang. Review of computational models for large-scale MDAO of urban air mobility concepts. In *AIAA SciTech 2024 Forum*, page 0377, 2024.

[145] R. A. Sauer, Z. Zou, and T. J. R. Hughes. A simple and efficient hybrid discretization approach to alleviate membrane locking in isogeometric thin shells. *arXiv preprint arXiv:2312.16944*, 2023.

[146] D. Schillinger, I. Harari, M.-C. Hsu, D. Kamensky, S. K. F. Stoter, Y. Yu, and Y. Zhao. The non-symmetric Nitsche method for the parameter-free imposition of weak boundary and coupling conditions in immersed finite elements. *Computer Methods in Applied Mechanics and Engineering*, 309:625–652, 2016.

[147] D. Schillinger and M. Ruess. The Finite Cell Method: A review in the context of higher-order structural analysis of CAD and image-based geometric models. *Archives of Computational Methods in Engineering*, 22(3):391–455, 2015.

[148] D. Schillinger, P. K. Ruthala, and L. H. Nguyen. Lagrange extraction and projection for NURBS basis functions: A direct link between isogeometric and standard nodal finite

element formulations. *International Journal for Numerical Methods in Engineering*, 108(6):515–534, 2016.

[149] R. Schmidt, J. Kiendl, K.-U. Bletzinger, and R. Wüchner. Realization of an integrated structural design process: analysis-suitable geometric modelling and isogeometric analysis. *Computing and Visualization in Science*, 13(7):315–330, Oct 2010.

[150] M. A. Scott, M. J. Borden, C. V. Verhoosel, T. W. Sederberg, and T. J. R. Hughes. Isogeometric finite element data structures based on Bézier extraction of T-splines. *International Journal for Numerical Methods in Engineering*, 88:126–156, 2011.

[151] L. Scotzniovsky, R. Xiang, Z. Cheng, G. Rodriguez, D. Kamensky, C. Mi, and J. T. Hwang. Geometric design of electric motors using adjoint-based shape optimization. *Optimization and Engineering*, pages 1–38, 2024.

[152] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph.*, 20(4):151–160, 1986.

[153] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. *ACM transactions on graphics (TOG)*, 22(3):477–484, 2003.

[154] F. Shirazian, R. Ghaffari, M. Hu, and R. A. Sauer. Hyperelastic material modeling of graphene based on density functional calculations. *PAMM*, 18(1):e201800419, 2018.

[155] J. Stegmann and E. Lund. Discrete material optimization of general composite shell structures. *International Journal for Numerical Methods in Engineering*, 62(14):2009–2027, 2005.

[156] H. Stolarski and T. Belytschko. Membrane locking and reduced integration for curved elements. *Journal of Applied Mechanics*, 49(1):172, 1982.

[157] K.Y. Sze, X.H. Liu, and S.H. Lo. Popular benchmark problems for geometric nonlinear analysis of shells. *Finite Elements in Analysis and Design*, 40(11):1551–1569, 2004.

[158] M.F.P. ten Eikelder, Y. Bazilevs, and I. Akkerman. A theoretical framework for discontinuity capturing: Joining variational multiscale analysis and variation entropy theory. *Computer Methods in Applied Mechanics and Engineering*, page 112664, 2019.

[159] D. C. Thomas, L. Engvall, S. K. Schmidt, K. Tew, and M. A. Scott. U-splines: Splines over unstructured meshes. *Computer Methods in Applied Mechanics and Engineering*, 401:115515, 2022.

[160] http://hangar.openvsp.org/. VSP hangar.

[161] https://github.com/david-kamensky/ShNAPr. ShNAPr source code.

[162] https://github.com/hanzhao2020/GOLDFISH. GOLDFISH source code.

[163] https://openmdao.org/newdocs/versions/latest/examples/beam_optimization_example.html. Optimizing the thickness distribution of a cantilever beam using the adjoint method.

[164] S. P. van Schie, M. Warner, A. Fletcher, and J. T. Hwang. Enforcing work conservation in modular aeroelastic coupling for multidisciplinary design optimization. In *AIAA SciTech 2024 Forum*, page 2411, 2024.

[165] S. P. van Schie, H. Zhao, J. Yan, R. Xiang, J. T. Hwang, and D. Kamensky. Solver-independent aeroelastic coupling for large-scale multidisciplinary design optimization. In *AIAA SciTech 2023 Forum*, page 0727, 2023.

[166] W. A. Wall, M. A. Frenzel, and C. Cyron. Isogeometric structural shape optimization. *Computer Methods in Applied Mechanics and Engineering*, 197(33-40):2976–2988, 2008.

[167] D. Wang and J.-S. Chen. Locking-free stabilized conforming nodal integration for meshfree Mindlin–Reissner plate formulation. *Computer Methods in Applied Mechanics and Engineering*, 193(12-14):1065–1083, 2004.

[168] Y. Wang, Y. Yu, and Y. Lin. Isogeometric analysis with embedded stiffened shells for the hull structural mechanical analysis. *Journal of Marine Science and Technology*, 27(1):786–805, 2022.

[169] R. Xiang, S. P. C. van Schie, L. Scotzniovsky, J. Yan, D. Kamensky, and J. T. Hwang. Automating adjoint sensitivity analysis for multidisciplinary models involving partial differential equations. *Structural and Multidisciplinary Optimization*, 67:1–31, 2024.

[170] F. Xu, E. L. Johnson, C. Wang, A. Jafari, C.-H. Yang, M. S. Sacks, A. Krishnamurthy, and M.-C. Hsu. Computational investigation of left ventricular hemodynamics following bioprosthetic aortic and mitral valve replacement. *Mechanics Research Communications*, 112:103604, 2021. Special issue honoring G.I. Taylor Medalist Prof. Arif Masud.

[171] F. Xu, S. Morganti, R. Zakerzadeh, D. Kamensky, F. Auricchio, A. Reali, T. J. R. Hughes, M. S. Sacks, and M.-C. Hsu. A framework for designing patient-specific bioprosthetic heart valves using immersogeometric fluid–structure interaction analysis. *International Journal for Numerical Methods in Biomedical Engineering*, 34(4):e2938, 2018.

[172] F. Xu, D. Schillinger, D. Kamensky, V. Varduhn, C. Wang, and M.-C. Hsu. The tetrahedral finite cell method for fluids: Immersogeometric analysis of turbulent flow around complex geometries. *Computers & Fluids*, 141:135–154, 2016. Advances in Fluid-Structure Interaction.

[173] J. Yan, N. Li, T. Luo, M. T. Tolley, and J. T. Hwang. Optimal control and design of an underactuated ball-pitching robotic arm using large-scale multidisciplinary optimization. In *AIAA Aviation 2019 Forum*, 2019.

[174] J. Yan, R. Xiang, D. Kamensky, M. T. Tolley, and J. T. Hwang. Topology optimization with automated derivative computation for multidisciplinary design problems. *Structural and Multidisciplinary Optimization*, 65(5):151, 2022.

[175] H. Yang, B. E. Abali, D. Timofeev, and W. H. Müller. Determination of metamaterial parameters by means of a homogenization approach based on asymptotic analysis. *Continuum mechanics and thermodynamics*, 32:1251–1270, 2020.

[176] Y. Yu, Y. Wang, and Y. Lin. Isogeometric analysis with non-conforming multi-patches for the hull structural mechanical analysis. *Thin-Walled Structures*, 187:110757, 2023.

[177] W. Zhang, T. Bui-Thanh, and M. S. Sacks. A machine learning approach for soft tissue remodeling. In *Proceedings of FEniCS'19*, 2019.

[178] H. Zhao, J. T. Hwang, and J.-S. Chen. Open-source shape optimization for isogeometric shells using FEniCS and OpenMDAO. *arXiv preprint arXiv:2410.02225*, 2024.

[179] H. Zhao, J. T. Hwang, and J.-S. Chen. Shape optimization of non-matching isogeometric shells with moving intersections. *Computer Methods in Applied Mechanics and Engineering*, 431:117322, 2024.

[180] H. Zhao, D. Kamensky, J. T. Hwang, and J.-S. Chen. Automated shape and thickness optimization for non-matching isogeometric shells using free-form deformation. *Engineering with Computers*, pages 1–24, 2024.

[181] H. Zhao, X. Liu, A. H. Fletcher, R. Xiang, J. T. Hwang, and D. Kamensky. An open-source framework for coupling non-matching isogeometric shells with application to aerospace structures. *Computers & Mathematics with Applications*, 111:109–123, 2022.