An open-source framework for coupling non-matching isogeometric shells with application to aerospace structures

Han Zhao[a], Xiangbei Liu[a], Andrew H. Fletcher[a], Ru Xiang[a], John T. Hwang[a], David Kamensky[a,*]

[a]*Department of Mechanical and Aerospace Engineering, University of California, San Diego, 9500 Gilman Drive, Mail Code 0411, La Jolla, CA 92093, USA*

Abstract

We introduce an open-source framework to directly analyze aerospace structures represented as collections of untrimmed NURBS patches, which is the representation used by NASA's Open Vehicle Sketch Pad (OpenVSP) for aircraft conceptual design. Airframes are modeled mechanically as Kirchhoff–Love shells and discretized isogeometrically for computational analysis. Coupling between separately-parameterized patches uses a slight modification of the penalty formulation proposed by Herrema et al. [A. J. Herrema, E. L. Johnson, D. Proserpio, M. C. H. Wu, J. Kiendl, M.C. Hsu, "Penalty coupling of non-matching isogeometric Kirchhoff–Love shell patches with application to composite wind turbine blades", Comput. Methods Appl. Mech. Engrg., 346 (2019) 810–840], which we verify using a similar suite of benchmark problems. Our open-source implementation leverages both advanced code generation through the FEniCS toolchain and efficient computational geometry operations through the Open Cascade modeling kernel. To demonstrate the framework's applicability to complicated industrial geometries, we perform stress analysis of the wing of an eCRM-002 electric vertical takeoff and landing (eVTOL) aircraft, with skin geometry designed in OpenVSP and internal stiffener geometry generated by an auxiliary tool. Source code for our non-matching shell analysis library PENGoLINS (PENalty-based GLuing of Isogeometric Non-matching Shells) will be maintained at https://github.com/hanzhao2020/PENGoLINS.

Keywords: Isogeometric analysis; FEniCS; Kirchhoff–Love shells; Non-matching interfaces; Aerospace structures

1. Introduction

Advances in battery technology, environmental concerns, urban congestion, and other factors have recently converged to produce a sudden growth of interest in electric vertical takeoff and landing (eVTOL) aircraft [1, 2]. Due to the scalability of electric motors (relative to internal

*Corresponding author
Email address: dmkamensky@eng.ucsd.edu (David Kamensky)

The final publication is available at Elsevier via https://doi.org/10.1016/j.camwa.2022.02.007

combustion engines used by traditional helicopters), a much wider variety of design configurations are up for consideration in this space, with no consensus on even high-level design features, such as the optimal number and placement of motors, batteries, lifting surfaces, etc. [3] This situation is in sharp contrast to helicopters and fixed-wing aircraft that use traditional modes of propulsion, which are mostly variations of mature, decades-old design templates. Thus, to explore the wide-open eVTOL design space, engineers will need new, more flexible workflows including ones that enable stress analysis on a diverse range of airframe geometries.

It is well-established that the analysis bottleneck in exploring geometrical design spaces is mesh generation [4]. The field of isogeometric analysis (IGA) [5, 6] aims to directly analyze spline-based geometry representations used in computer-aided design (CAD) programs, bypassing mesh generation entirely. 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 [7], and the additional regularity permits Bubnov–Galerkin approximations of high-order PDEs.

A particular high-order PDE model relevant to aerospace structural analysis is Kirchhoff–Love shell theory. IGA of Kirchhoff–Love shells has been studied extensively by Kiendl and various collaborators [8–14], and has been applied with excellent results across a variety of domains, including analysis of wind turbine blades [15–18], heart valve leaflets [19–23], and graphene sheets [24–26]. The basic idea of using a Bubnov–Galerkin method for the problem was limited to fully C^1 spline spaces, but could be extended to multi-patch splines with matching parameterizations via the bending strip method [27] or constraint equations [28]. Designs consisting of multiple non-matching and/or trimmed spline patches have been analyzed using a variety of methods, including Nitsche-type coupling [29, 30], penalty methods [31], and super-penalty methods [32, 33].

In the present work, we adapt the penalty coupling method of [31] for use with a spline-based geometry framework for conceptual design of airframes of eVTOL and other types of aircraft. This framework combines NASA's Open Vehicle Sketch Pad (OpenVSP) [34–37] and custom software for specifying the layout of internal stiffeners within wings. The eventual aim of this work is to use the proposed analysis framework for design optimization. In that light, another important contribution of the present work is its use of automated code generation, via the FEniCS Project [38] and its extension for IGA, called tIGAr [39]. This allows for efficient, automated access to derivative information needed by gradient-based optimization algorithms.

We begin, in Section 2, by explaining the formulation used for non-matching multi-patch IGA of shells. Section 3 then describes our FEniCS- and tIGAr-based implementation of this method, which is available as the open-source library PENGoLINS (PENalty-based GLuing of Isogeomet-

ric Non-matching Shells) [40].[1] Section 4 verifies that PENGoLINS accurately solves a collection of linear and nonlinear benchmark problems from the literature, before Section 5 demonstrates its application to stress analysis of an eVTOL wing with internal structural members. Lastly, Section 6 draws conclusions and outlines plans for future research.

2. Isogeometric shell formulation with penalty-based coupling

We model aircraft structural components as Kirchhoff–Love shells. A Kirchhoff–Love shell with a C^1 or smoother spline geometry can be conveniently discretized using isogeometric analysis, as the spline function space is sufficiently regular to directly apply a conforming Galerkin method to the fourth-order governing equation. We refer the reader to earlier work by Kiendl and collaborators [8, 9] for the formulation used throughout this paper, where we assume a St. Venant–Kirchhoff constitutive model for the shell structure. (Generalizations to arbitrary hyperelastic materials can be found in [10, 41].)

While we leave the full Kirchhoff–Love shell formulation to the cited references, we review some basic kinematic definitions to establish notation for the remainder of this paper. The geometry of a shell structure component associated with each single-patch structural component can be described mathematically as a parametric surface $\mathbf{X}(\xi^1, \xi^2)$ representing the shell's midsurface, where ξ^1 and ξ^2 are parametric coordinates. The deformed configuration of the shell structure's midsurface is then given by

$$\mathbf{x}(\xi^1, \xi^2) = \mathbf{X}(\xi^1, \xi^2) + \mathbf{u}(\xi^1, \xi^2) \,, \tag{1}$$

where $\mathbf{u}$ is the midsurface displacement field. Kirchhoff–Love kinematic assumptions imply that the deformation of material away from the midsurface is uniquely determined by $\mathbf{u}$. Covariant basis vectors for the tangent spaces of $\mathbf{x}$ and $\mathbf{X}$ are defined by

$$\mathbf{a}_\alpha = \frac{\partial \mathbf{x}}{\partial \xi^\alpha} \quad \text{and} \quad \mathbf{A}_\alpha = \frac{\partial \mathbf{X}}{\partial \xi^\alpha} \quad \text{for } \alpha = 1, 2 \,, \tag{2}$$

along with unit normal vectors

$$\mathbf{a}_3 = \frac{\mathbf{a}_1 \times \mathbf{a}_2}{\|\mathbf{a}_1 \times \mathbf{a}_2\|} \quad \text{and} \quad \mathbf{A}_3 = \frac{\mathbf{A}_1 \times \mathbf{A}_2}{\|\mathbf{A}_1 \times \mathbf{A}_2\|} \,, \tag{3}$$

where $\|\cdot\|$ is the Euclidean norm. Corresponding local orthonormal bases $\{\mathbf{e}_i\}_{i=1}^3$ and $\{\mathbf{E}_i\}_{i=1}^3$ are then defined by applying the Gram–Schmidt procedure to $\{\mathbf{a}_i\}$ and $\{\mathbf{A}_i\}$.

[1] The "o" is added for pronunciation, like the "ni" in FEniCS. "Pengolin" is an archaic spelling of "pangolin". The pangolin is an anteater-like mammal that is most conspicuous for its armored exterior, consisting of many small shells coupled together.

To couple separately-parameterized shell structures along their intersection curves, we augment the formulation of [8] with an additional penalty method, borrowing heavily from [31]. In particular, for each intersection curve $\mathcal{L}$ between NURBS patches A and B, we define a penalty energy of the form

$$W_{\text{pen}}^{AB} = \frac{1}{2}\int_{\mathcal{L}} \alpha_{\text{d}} \left\|\mathbf{u}^A - \mathbf{u}^B\right\|^2 + \alpha_{\text{r}}\left(\left(\mathbf{e}_3^A \cdot \mathbf{e}_3^B - \mathbf{E}_3^A \cdot \mathbf{E}_3^B\right)^2 + \left(\left\|\pi^A \mathbf{e}_3^B\right\| - \left\|\Pi^A \mathbf{E}_3^B\right\|\right)^2\right) d\mathcal{L}\,, \tag{4}$$

where superscripts A and B indicate quantities evaluated on the corresponding surfaces, $\alpha_{\text{d}} > 0$ and $\alpha_{\text{r}} > 0$ are penalty parameters defined by formulas given in [31, Section 2.4], and π^A and Π^A are Euclidean orthogonal projections onto the tangent space of surface A in the deformed and original configurations, i.e.,

$$\pi^A \mathbf{v} = \left(\mathbf{v} \cdot \mathbf{e}_\alpha^A\right)\mathbf{e}_\alpha^A \quad \text{and} \quad \Pi^A \mathbf{V} = \left(\mathbf{V} \cdot \mathbf{E}_\alpha^A\right)\mathbf{E}_\alpha^A\,, \tag{5}$$

in which the index α has range $\{1, 2\}$ and is summed over when repeated. In the formulation of the full, coupled multi-patch shell problem, the energy (4) for each intersection curve is added to the sum of internal elastic potential energies associated with the deformations of the shell patches.

The α_{d} term of (4) is taken directly from [31], although the cited reference bypasses the energetic interpretation, and directly states the Gateaux derivative $\delta W_{\text{pen}}^{AB}$ of its penalty energy with respect to displacement; this formulation is provided for comparison in Remark 1. To interpret the α_{r} term, note that $\|\pi^A \mathbf{e}_3^B\|$ is the cosine of the angle between the normal vector of surface B and a tangent plane to surface A in the deformed configuration, and similarly for $\|\Pi^A \mathbf{E}_3^B\|$ in the original configuration. Thus, the formulation penalizes changes in angle between surfaces A and B. The corresponding term from [31] penalizes changes in the angle between the normal of surface B and a conormal vector that is tangent to surface A and orthogonal to $\mathcal{L}$. When compared with the formulation of [31], the α_{r} term of (4) penalizes additional relative rotations about the conormal. However, such out-of-plane "tearing" modes are already suppressed strongly by the α_{d} term, so we expect no significant difference in results. For the hinge-like modes of primary concern, (4) is equivalent to the formulation of [31]. Our sole reason for modifying the formulation of [31] is that (4) is more convenient to implement in the case where $\mathcal{L}$ is an arbitrary surface–surface intersection and the conormal vector used by [31] becomes difficult to obtain in symbolic form. (The problems considered in [31] are limited to cases where patches are coupled at edges.)

We use the formulas for α_{d} and α_{r} from [31, Section 2.4], which are both proportional to a single dimensionless scalar $\alpha > 0$. We refer to α here as the "(dimensionless) penalty coefficient". For an isotropic, homogeneous St. Venant–Kirchhoff material, we use

$$\alpha_{\text{d}} = \frac{\alpha E t}{h(1-\nu^2)} \quad \text{and} \quad \alpha_{\text{r}} = \frac{\alpha E t^3}{12h(1-\nu^2)}\,, \tag{6}$$

where E and ν are the Young's modulus and Poisson ratio, t is the shell thickness, and h is a measure of element size. (More elaborate formulas for multi-layer composite shells are given in the cited reference.) On nonuniform and anisotropic meshes, the element size "h" can be ambiguous. In this work, we take $h = \frac{1}{2}\left(h_X^A + h_X^B\right)$ with

$$h_X = h_\xi \sqrt{\mathrm{tr}\left(\frac{\partial \mathbf{X}}{\partial \boldsymbol{\xi}} \cdot \frac{\partial \mathbf{X}}{\partial \boldsymbol{\xi}}^T\right)}, \tag{7}$$

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 [31] reports averaging "the lengths of the local elements in the direction most parallel to the penalty curve" [31, page 818] to obtain "h", but we prefer the isotropic definition (7) 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 [31] 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 4, producing results similar to those of [31], despite the minor differences in formulation and entirely disjoint software implementations.

Remark 1. This remark restates the formulation of [31], for comparison purposes; to be clear, it is *not* what is used in the present work. Reference [31] enforces displacement and rotation continuity with a penalty term of the form

$$\begin{aligned}\delta \widetilde{W}_{\mathrm{pen}}^{AB} = \int_{\mathcal{L}} & \alpha_{\mathrm{d}} \left(\mathbf{u}^A - \mathbf{u}^B\right) \cdot \left(\delta \mathbf{u}^A - \delta \mathbf{u}^B\right) \\ & + \alpha_{\mathrm{r}} \left(\left(\mathbf{a}_3^A \cdot \mathbf{a}_3^B - \mathbf{A}_3^A \cdot \mathbf{A}_3^B\right)\left(\delta \mathbf{a}_3^A \cdot \mathbf{a}_3^B + \mathbf{a}_3^A \cdot \delta \mathbf{a}_3^B\right)\right. \\ & \left. + \left(\mathbf{a}_n^A \cdot \mathbf{a}_3^B - \mathbf{A}_n^A \cdot \mathbf{A}_3^B\right)\left(\delta \mathbf{a}_n^A \cdot \mathbf{a}_3^B + \mathbf{a}_n^A \cdot \delta \mathbf{a}_3^B\right)\right) d\mathcal{L}\,,\end{aligned} \tag{8}$$

where "δ" denotes a Gateaux derivative with respect to displacement in the direction $\delta\mathbf{u}$, the vector $\mathbf{a}_n^A$ is the unit conormal tangent to surface A and normal to $\mathcal{L}$ in the deformed configuration, and $\mathbf{a}_n^B$, $\mathbf{A}_n^A$, and $\mathbf{A}_n^B$ are defined analogously, but for surface B and/or in the reference configuration.

Remark 2. Coradello et al. [32] recently introduced a super-penalty method involving a carefully-designed projection of the penalty forces. This projected super-penalty method exhibits optimal convergence rates in cases where patches are coupled at their edges or at trim curves. In a subsequent paper, Coradello et al. [33] found that, in benchmark tests with exact solutions, the method

[2] For technical reasons, (7) 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.

of [31] exhibits sub-optimal convergence rates, but this was only "noticeable in the asymptotic regime for $p = 4$" [33, page 16], and is thus unlikely to meaningfully impact aerospace structural analysis, where modeling error dominates well before that point. It would be trivial to modify PENGoLINS to use a super-penalty method without projection, but even the projected super-penalty method will no longer be optimal for arbitrary surface–surface intersections, unless patches are split into multiple trimmed surfaces at each intersection, which is outside the scope of our targeted OpenVSP-based geometry pipeline. Thus, we prefer the original low-order penalty method of [31], as it is sufficiently accurate for aerospace structures at practical resolutions and does not require specialized projection techniques and preconditioners to contend with the rapid blow-up of super-penalties under mesh refinement. Further, the method of [31] was found to better approximate rotational constraints on coarse meshes [33, Figure 12a].

3. Implementation using FEniCS and tIGAr

This section introduces the design and workflow of PENGoLINS, an open-source Python library that interacts with FEniCS [38] and tIGAr [39] 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) [42] for specifying variational forms, which can be automatically compiled [43] into low-level numerical routines for use with the C++ finite element library DOLFIN [44].

The library tIGAr extends FEniCS to IGA, using the concept of extraction [45–47] to represent IGA in terms of traditional finite element operations. In particular, tIGAr represents a piecewise polynomial spline space $\mathcal{V}^{\text{IGA}}$ through an extraction matrix $\mathbf{M}$. Each column of $\mathbf{M}$ defines a basis function from $\mathcal{V}^{\text{IGA}}$ by providing its coefficients in a linear combination of finite element basis functions from the Lagrange finite element space $\mathcal{V}^{\text{FE}}$. Thus, linear and bilinear forms specified in UFL can be assembled using FEniCS's automated workflow to obtain the vector $\mathbf{F}^{\text{FE}}$ and matrix $\mathbf{K}^{\text{FE}}$, giving coordinate representations of these forms in $\mathcal{V}^{\text{FE}}$. The extraction operator $\mathbf{M}$ is then used to obtain $\mathbf{F}^{\text{IGA}} = \mathbf{M}^T\mathbf{F}^{\text{FE}}$ and $\mathbf{K}^{\text{IGA}} = \mathbf{M}^T\mathbf{K}^{\text{FE}}\mathbf{M}$, to produce a linear system which can be solved for IGA degrees of freedom (DoFs). A more detailed technical explanation is given in [39].

Our starting point for tIGAr-based Kirchhoff–Love shell analysis is the existing open-source library ShNAPr [48], which was originally developed for [49]. 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 [50]. 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.

3.1. 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.

3.1.1. 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 [51], a geometry modeler for unconventional aircraft configurations. We use pythonOCC [52], 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 3. 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 3.2.

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 [53], 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., [54, Section A.2.2] and [55, Figures 5 and 6], where effectively high-order convergence is obtained with similar low-order boundary quadrature, or [56, 57], 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 "mortar mesh". The parametric coordinates of mortar 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.

3.1.2. Assembling the penalty terms for one intersection curve

We now walk through the process of assembling penalty terms on the intersection curve $\mathcal{L}$ between two patches, S^A and S^B, as illustrated by Figure 1, in both physical space and spline parameter space. As discussed in Section 3.1.1, the preprocessing stage of the analysis creates a mortar mesh of $\mathcal{L}$ 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 mortar mesh.

Remark 4. 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 (4), using the `derivative` function. Thus, only the energy (4) is directly specified in UFL; its associated virtual work and linearization are derived automatically.

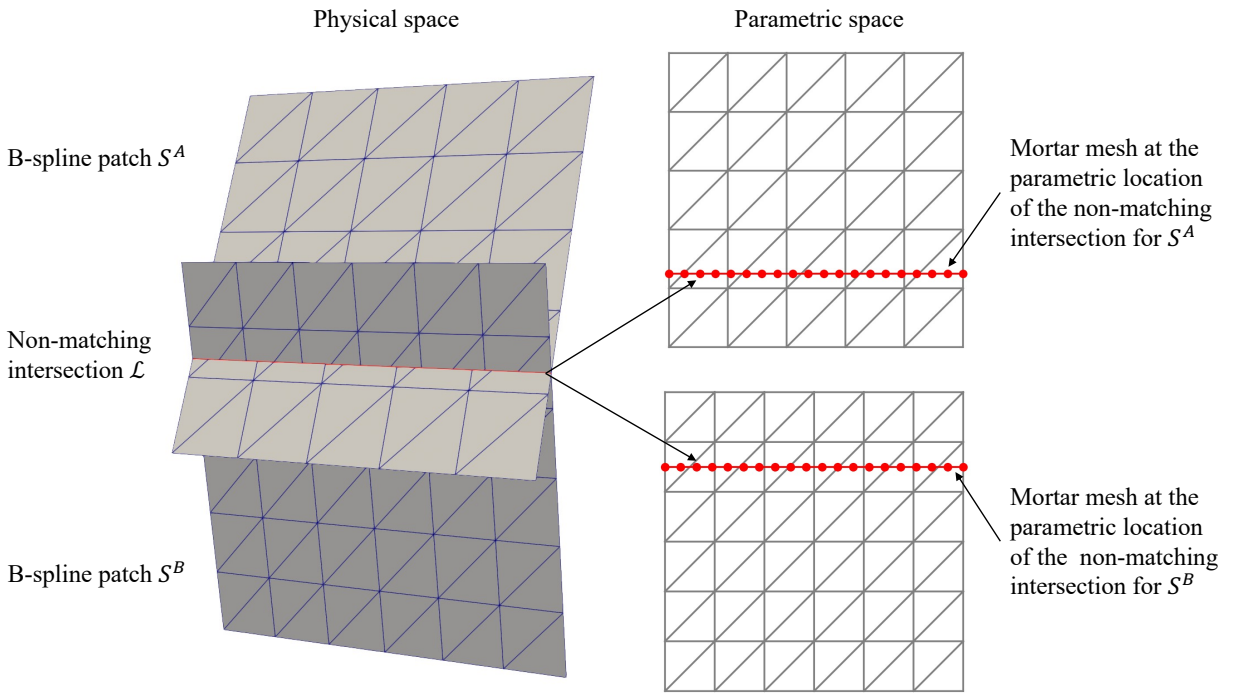


Figure 1: Schematic configuration of spline patches and mortar mesh used for penalty quadrature. Patches are tessellated into triangles for technical reasons (cf. Remark 6).

Remark 5. Since the α_r term of (4) involves the first derivatives of functions from the spline patches, we must also interpolate these derivatives at quadrature points of the mortar mesh. This involves a modified version of `PETScDMCollection`, which is included within PENGoLINS.

Remark 6. 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 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 $\mathcal{V}^{A,\mathrm{FE}}$ and $\mathcal{V}^{B,\mathrm{FE}}$ with DoFs for the finite element (FE) representation of IGA spline spaces on patches S^A and S^B.

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

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

4. Move Ω^M to configuration $\widehat{\Omega}^{M,A}$ and use `PETScDMCollection` (modified as discussed in Remark 5) to generate an interpolation matrix $\mathbf{A}^A$ from $\mathcal{V}^{A,\mathrm{FE}}$ to $\mathcal{V}^M$. Then move Ω^M to $\widehat{\Omega}^{M,B}$ and likewise generate $\mathbf{A}^B$.[3]

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

6. Use FEniCS to specify the penalty energy (4) in UFL and compile kernels for assembling its Gateaux derivatives (obtained automatically via the UFL `derivative` function) over Ω^M. Then assemble the vectors $\mathbf{F}^{A,M}$ and $\mathbf{F}^{B,M}$, corresponding to the $\mathcal{V}^M$ coordinate representations of the parts of the form $-\delta W^{AB}_{\mathrm{pen}}$ that are linear in test functions from patches A and B. Likewise, assemble the matrices $\mathbf{K}^{AA,M}$, $\mathbf{K}^{AB,M}$, $\mathbf{K}^{BA,M}$ and $\mathbf{K}^{BB,M}$, corresponding to the Jacobians of $\mathbf{F}^{A,M}$ and $\mathbf{F}^{B,M}$ with respect to the DoFs in $\mathcal{V}^M$ used to interpolate displacements on each patch.

7. Lastly, transfer $\mathbf{F}^{A,M}$ and $\mathbf{F}^{B,M}$ and $\mathbf{K}^{AA,M}$, $\mathbf{K}^{AB,M}$, $\mathbf{K}^{BA,M}$ and $\mathbf{K}^{BB,M}$ to vectors and matrices corresponding to DoFs of $\mathcal{V}^{A,\mathrm{FE}}$ and $\mathcal{V}^{B,\mathrm{FE}}$. This is accomplished via matrix-vector and matrix-matrix products, using the interpolation matrices and their transposes, e.g., $\mathbf{F}^{A,\mathrm{FE}} = (\mathbf{A}^A)^T\mathbf{F}^{A,M}$, $\mathbf{K}^{AB,\mathrm{FE}} = (\mathbf{A}^A)^T\mathbf{K}^{AB,M}\mathbf{A}^B$, 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 3.2.

3.2. 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 $\{S^1, S^2, \ldots, S^m\}$ that have n non-matching intersections $\{\mathcal{L}^1, \mathcal{L}^2, \ldots, \mathcal{L}^n\}$ 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 (4) for each intersection can be

[3] As mentioned in Remark 5, we interpolate not just functions but their derivatives. The space $\mathcal{V}^M$ 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.

written as follows: Find displacements of m shells $\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^m \in \mathcal{V}^1, \mathcal{V}^2, \ldots, \mathcal{V}^m$ such that for all $\mathbf{v}^1, \mathbf{v}^2, \ldots, \mathbf{v}^m \in \mathcal{V}^1, \mathcal{V}^2, \ldots, \mathcal{V}^m$,

$$\begin{aligned}\delta W = {} & \delta W_s^1(\mathbf{u}^1, \mathbf{v}^1) + \delta W_s^2(\mathbf{u}^2, \mathbf{v}^2) + \ldots + \delta W_s^m(\mathbf{u}^m, \mathbf{v}^m) \\ & + \delta W_{\text{pen}}^{i^{11} i^{12}}(\mathbf{u}^{i^{11}}, \mathbf{u}^{i^{12}}, \mathbf{v}^{i^{11}}, \mathbf{v}^{i^{12}}) + \delta W_{\text{pen}}^{i^{21} i^{22}}(\mathbf{u}^{i^{21}}, \mathbf{u}^{i^{22}}, \mathbf{v}^{i^{21}}, \mathbf{v}^{i^{22}}) \\ & + \ldots + \delta W_{\text{pen}}^{i^{n1} i^{n2}}(\mathbf{u}^{i^{n1}}, \mathbf{u}^{i^{n2}}, \mathbf{v}^{i^{n1}}, \mathbf{v}^{i^{n2}}) = 0 \,, \end{aligned} \tag{9}$$

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 $\mathcal{V}^i$ is the space of displacements on patch i.

In a slight abuse of notation, we shall also use $\mathbf{u}^i$ to refer to the vector of DoFs of the space $\mathcal{V}^i$. Then the variational problem (9) gives rise to the following system of nonlinear algebraic equations:

$$\frac{\partial W}{\partial \mathbf{u}^1} = \mathbf{0} \,, \tag{10}$$

$$\frac{\partial W}{\partial \mathbf{u}^2} = \mathbf{0} \,, \tag{11}$$

$$\ldots \,,$$

$$\frac{\partial W}{\partial \mathbf{u}^m} = \mathbf{0} \,. \tag{12}$$

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

$$\begin{bmatrix} \dfrac{\partial^2 W}{(\partial \mathbf{u}^1)^2} & \dfrac{\partial^2 W}{\partial \mathbf{u}^1 \partial \mathbf{u}^2} & \cdots & \dfrac{\partial^2 W}{\partial \mathbf{u}^1 \partial \mathbf{u}^m} \\ \dfrac{\partial^2 W}{\partial \mathbf{u}^2 \partial \mathbf{u}^1} & \dfrac{\partial^2 W}{(\partial \mathbf{u}^2)^2} & \cdots & \dfrac{\partial^2 W}{\partial \mathbf{u}^2 \partial \mathbf{u}^m} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 W}{\partial \mathbf{u}^m \partial \mathbf{u}^1} & \dfrac{\partial^2 W}{\partial \mathbf{u}^m \partial \mathbf{u}^2} & \cdots & \dfrac{\partial^2 W}{(\partial \mathbf{u}^m)^2} \end{bmatrix} \begin{bmatrix} \Delta \mathbf{u}^1 \\ \Delta \mathbf{u}^2 \\ \vdots \\ \Delta \mathbf{u}^m \end{bmatrix} = \begin{bmatrix} -\dfrac{\partial W}{\partial \mathbf{u}^1} \\ -\dfrac{\partial W}{\partial \mathbf{u}^2} \\ \vdots \\ -\dfrac{\partial W}{\partial \mathbf{u}^m} \end{bmatrix} , \tag{13}$$

where the left hand side (LHS) of (13) 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,

$$\frac{\partial W}{\partial \mathbf{u}^i} = \frac{\partial W_s^i}{\partial \mathbf{u}^i} + \frac{\partial W_{\text{pen}}^{i j_1}}{\partial \mathbf{u}^i} + \frac{\partial W_{\text{pen}}^{i j_2}}{\partial \mathbf{u}^i} + \ldots + \frac{\partial W_{\text{pen}}^{i j_n}}{\partial \mathbf{u}^i} \,, \tag{14}$$

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 (14), which is the entry (i, j) of LHS of (13)

$$\frac{\partial^2 W}{\partial \mathbf{u}^i \partial \mathbf{u}^j} = \begin{cases} \dfrac{\partial^2 W_s^i}{(\partial \mathbf{u}^i)^2} + \dfrac{\partial^2 W_{\text{pen}}^{ij_1}}{(\partial \mathbf{u}^i)^2} + \dfrac{\partial^2 W_{\text{pen}}^{ij_2}}{(\partial \mathbf{u}^i)^2} + \ldots + \dfrac{\partial^2 W_{\text{pen}}^{ij_n}}{(\partial \mathbf{u}^i)^2} , & j = i \\ \dfrac{\partial^2 W_{\text{pen}}^{ij}}{\partial \mathbf{u}^i \partial \mathbf{u}^j} , & j \in \{j_1, j_2, \ldots, j_n\} \\ \mathbf{0} , & \text{otherwise} \end{cases} . \tag{15}$$

Substituting (14) and (15) into equation (13), 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

$$\begin{bmatrix} \mathbf{K}^{11,\text{FE}} & \mathbf{K}^{12,\text{FE}} & \ldots & \mathbf{K}^{1m,\text{FE}} \\ \mathbf{K}^{21,\text{FE}} & \mathbf{K}^{22,\text{FE}} & \ldots & \mathbf{K}^{2m,\text{FE}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{K}^{m1,\text{FE}} & \mathbf{K}^{m2,\text{FE}} & \ldots & \mathbf{K}^{mm,\text{FE}} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{u}^{1,\text{FE}} \\ \Delta\mathbf{u}^{2,\text{FE}} \\ \vdots \\ \Delta\mathbf{u}^{m,\text{FE}} \end{bmatrix} = \begin{bmatrix} \mathbf{F}^{1,\text{FE}} \\ \mathbf{F}^{2,\text{FE}} \\ \vdots \\ \mathbf{F}^{m,\text{FE}} \end{bmatrix} . \tag{16}$$

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 3:

$$\begin{bmatrix} \mathbf{K}^{11,\text{IGA}} & \mathbf{K}^{12,\text{IGA}} & \ldots & \mathbf{K}^{1m,\text{IGA}} \\ \mathbf{K}^{21,\text{IGA}} & \mathbf{K}^{22,\text{IGA}} & \ldots & \mathbf{K}^{2m,\text{IGA}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{K}^{m1,\text{IGA}} & \mathbf{K}^{m2,\text{IGA}} & \ldots & \mathbf{K}^{mm,\text{IGA}} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{u}^{1,\text{IGA}} \\ \Delta\mathbf{u}^{2,\text{IGA}} \\ \vdots \\ \Delta\mathbf{u}^{m,\text{IGA}} \end{bmatrix} = \begin{bmatrix} \mathbf{F}^{1,\text{IGA}} \\ \mathbf{F}^{2,\text{IGA}} \\ \vdots \\ \mathbf{F}^{m,\text{IGA}} \end{bmatrix} , \tag{17}$$

where

$$\mathbf{K}^{ij,\text{IGA}} = \left(\mathbf{M}^i\right)^T \mathbf{K}^{ij,\text{FE}} \mathbf{M}^j \quad , \quad \mathbf{F}^{i,\text{IGA}} = \left(\mathbf{M}^i\right)^T \mathbf{F}^{i,\text{FE}} \tag{18}$$

(*without* summation on the repeated indices i and j), and $\{\Delta\mathbf{u}^{i,\text{IGA}}\}$ 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 mortar 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 3.1.2, based on the list of mappings `mapping_list` and the mortar 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 (16). The block LHS matrix is represented using the `MatNest` abstraction from the PETSc [58–60] 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 (17) and solves it using either a direct or iterative solver.

4. Benchmark tests

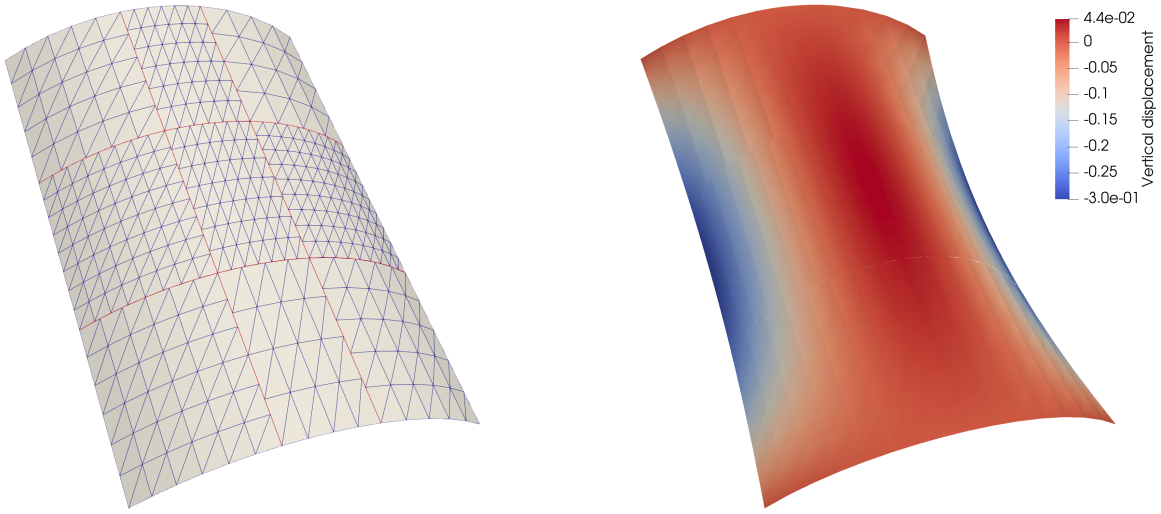
This section demonstrates the use of PENGoLINS to solve several benchmark problems from the literature. Each problem is selected to verify specific functionality from PENGoLINS. Unless otherwise specified, results are obtained using the default value of $\alpha = 10^3$ for the dimensionless penalty coefficient.

4.1. Scordelis–Lo roof

The first benchmark we consider is the Scordelis–Lo roof example [61], where we divide its geometry into nine separately-parameterized NURBS surfaces, as depicted in Figure 2a.[4] 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

[4]Triangular elements are used for extraction and visualization, for the reasons discussed in Remark 6.

provided in [31, Section 3.1]. The computed displacement in the vertical direction for geometry in Figure 2a is shown in Figure 2b. This plot clearly indicates that our framework is able to maintain approximate displacement continuity on the non-matching interfaces during deformation.



(a) (b)

Figure 2: (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.

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 NURBS surfaces in Figure 3. The converged solution of the QoI is reported in [9, 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 2b corresponds to the first data point of cubic NURBS surfaces in Figure 3.

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 4. 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 [31] 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 5. Methods for stress resultant computation and stress recovery in isogeometric Kirchhoff–Love shells are detailed by [9, Section 3.4],

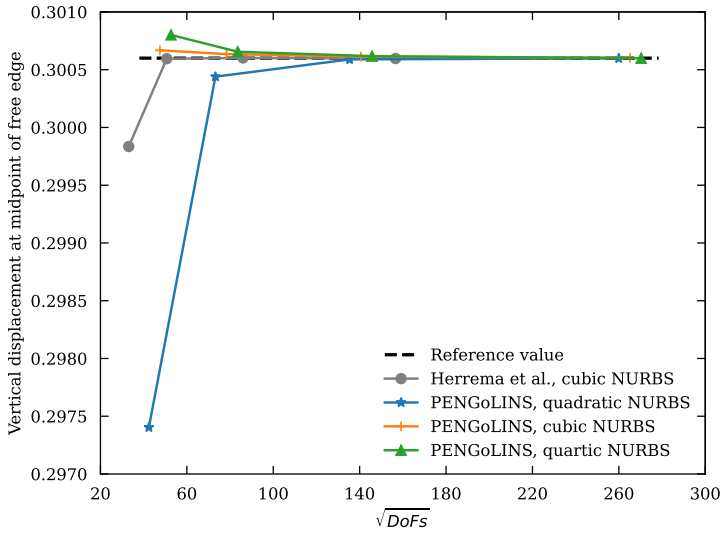


Figure 3: Convergence of Scordelis–Lo roof example for different orders of NURBS surfaces. Results from [31] (using a different mesh structure) are included for reference.

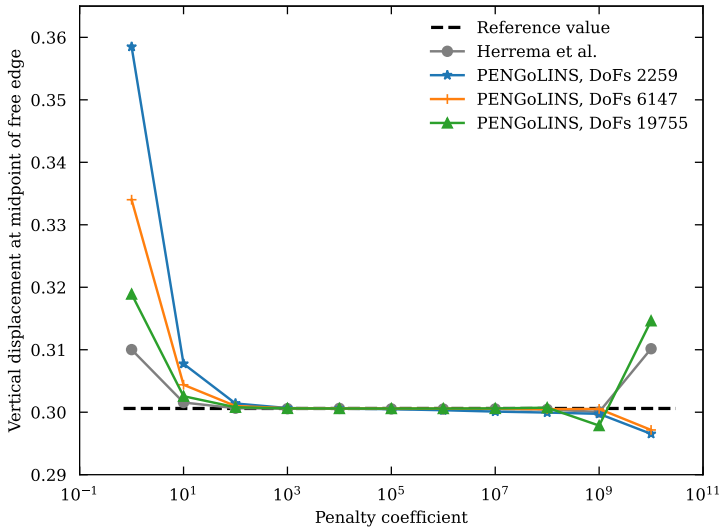
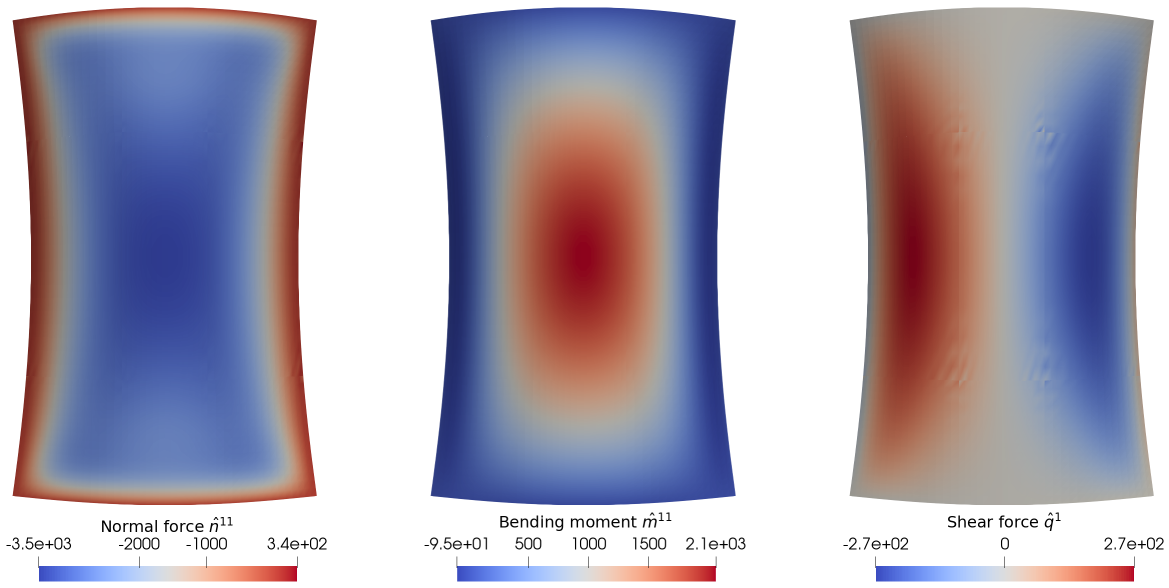


Figure 4: A wide range of penalty coefficients compute accurate results for Scordelis–Lo roof example. Results from [31, Figure 6(a), non-matching] are included for reference.

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 [9, Section 6.2.4]. The resultant distributions shown in Figure 5 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 1, also demonstrating good agreement, especially of maximum absolute values, which are most relevant to design.

Remark 7. Instability in transverse shear resultants at intersections is not surprising, because these resultants are obtained by directly evaluating *third* derivatives of the displacement [9, (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 5, but the penalty formulation only (approximately) enforces C^1 continuity at intersections.



(a) (b) (c)

Figure 5: 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 $\hat{q}^1$.

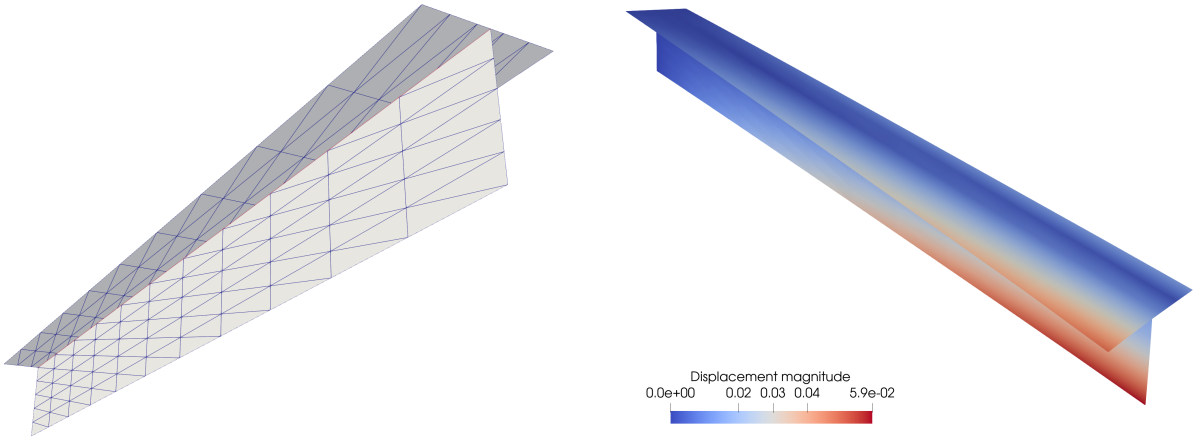
Stress resultants	Non-matching shell analysis		Abaqus shell analysis		Single patch shell analysis	
	Min	Max	Min	Max	Min	Max
$\hat{n}^{11}$	-3487	336	-3417	127	-3510	25
$\hat{m}^{11}$	-95	2055	-93	2079	-91	2053
$\hat{q}^1$	-273	273	-278	278	-280	280

Table 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 [9, Section 6.2.4].

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 depicted in

Figure 6a, 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 [31, Section 3.3]. However, unlike the problem setup in [31], 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 6b.

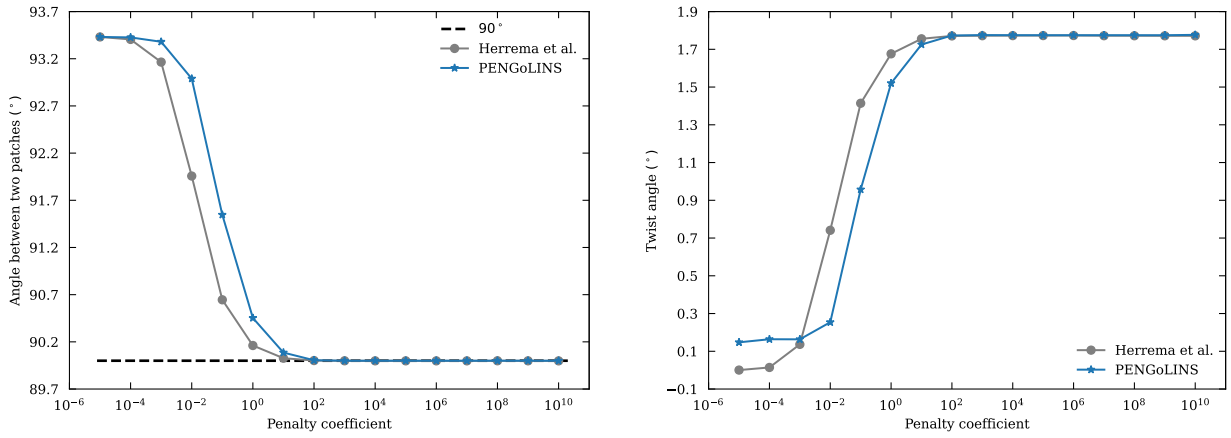


(a) (b)

Figure 6: (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 7a. 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 7b, 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 [31, Figure 15]. Thus, Figure 7 demonstrates that the recommended penalty coefficient of 10^3 identified in Section 4.1 remains effective at preserving rotational continuity when patches meet at a nonzero angle in the initial configuration. Comparing with the results of [31] 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 (cf. Section 2), but, even with these differences, results are practically indistinguishable

for α around or beyond the recommended value of 10^3.



(a) (b)

Figure 7: (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 [31] (using a different mesh and definition of element size) are included for reference.

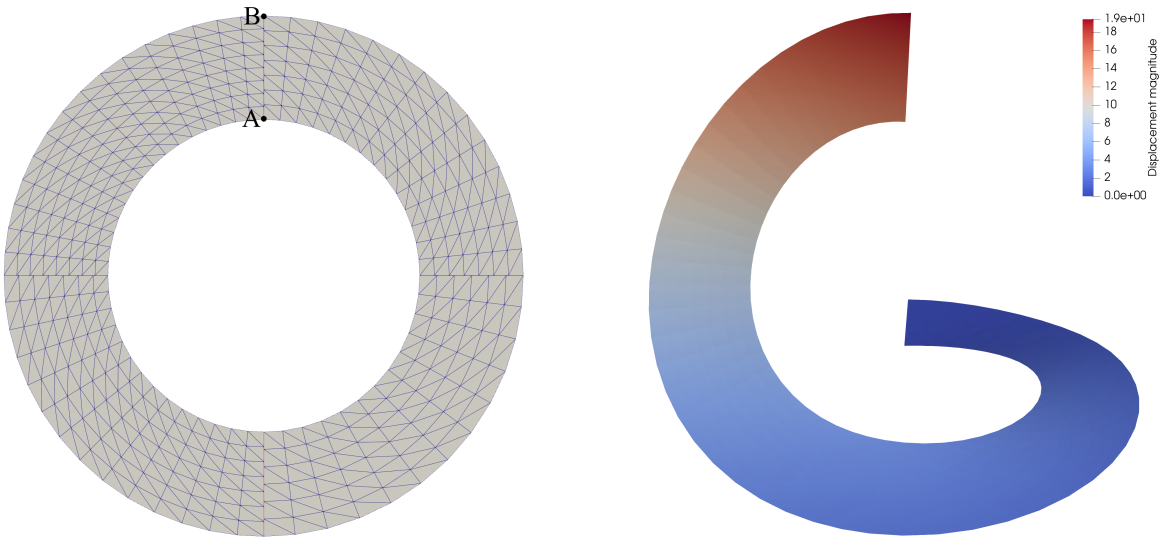
4.3. Nonlinear analysis of a slit annular plate

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. [62], 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 8a. Figure 8b shows the resulting deformed configuration at the maximum load. Displacement remains qualitatively smooth across the non-matching interfaces.

We compare the vertical displacements of two points, labelled A and B in Figure 8a, with reference values provided in [62, Table 4] and benchmark results reported in [31, Figure 23]. Figure 9 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.

4.4. Curved intersections

In the benchmark tests of Section 4.1–4.3, the intersection curves between patches are parallel to coordinate lines of one spline parameter and orthogonal to coordinate lines of the other. This



(a) (b)

Figure 8: (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.

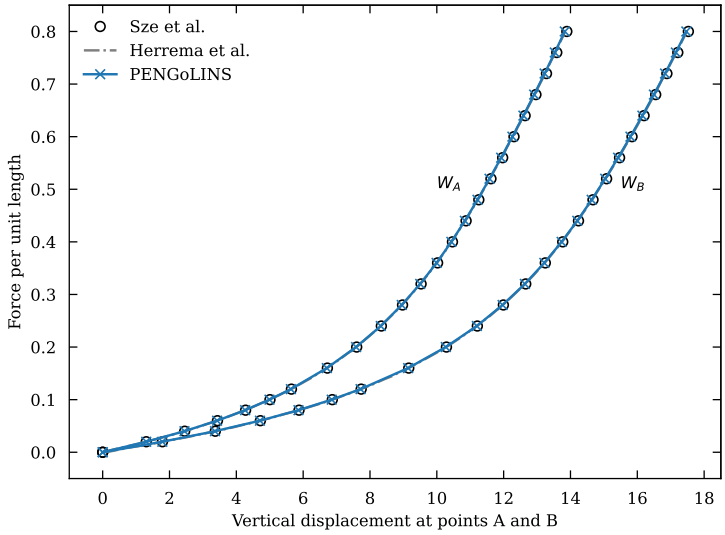


Figure 9: Comparison between our computations and reference data for vertical displacement at points A and B.

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 10, such that the intersection is no longer orthogonal (in physical space) to parametric coordinate lines running in the axial direction. Figure 11 compares the convergence of displacement using this parameterization with the results from Section 4.1. The formulation clearly still converges rapidly under

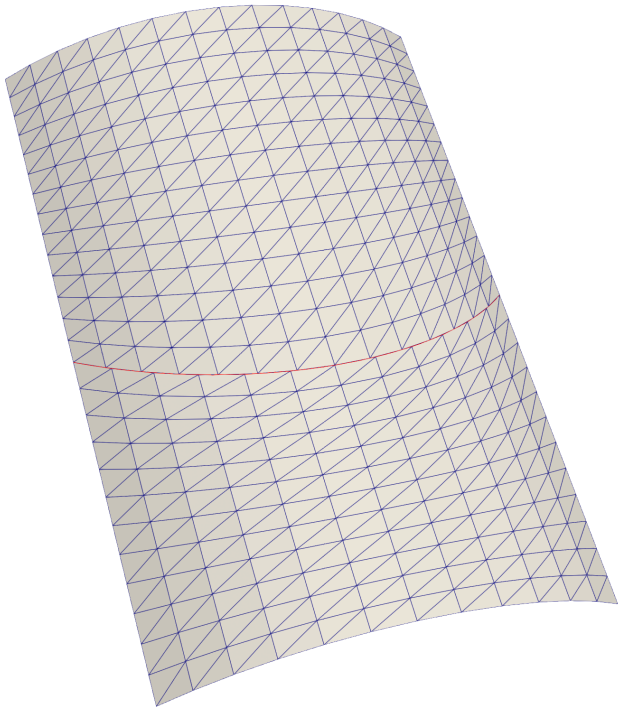


Figure 10: Distorted parameterization of the Scordelis–Lo roof.

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

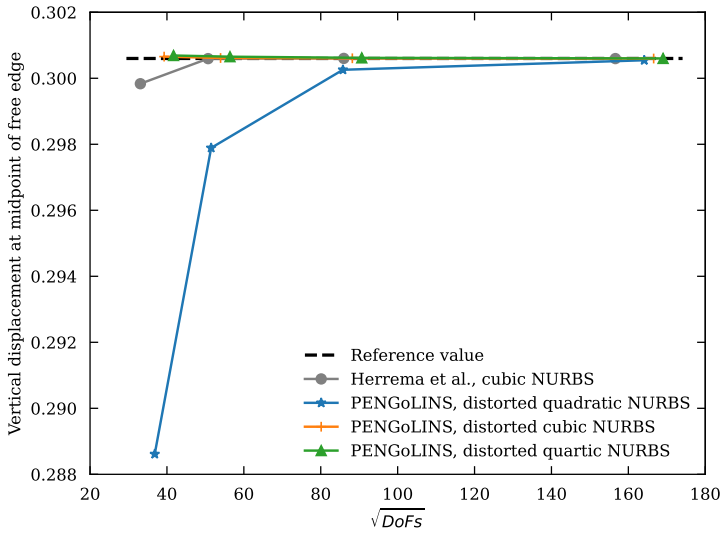


Figure 11: Convergence of displacement in the Scordelis–Lo roof with a distorted parameterization.

parameterization of the top patch as shown in Figure 12, such that the intersection becomes curved in the spline parameter space. Figure 13 demonstrates that the angular constraint between the

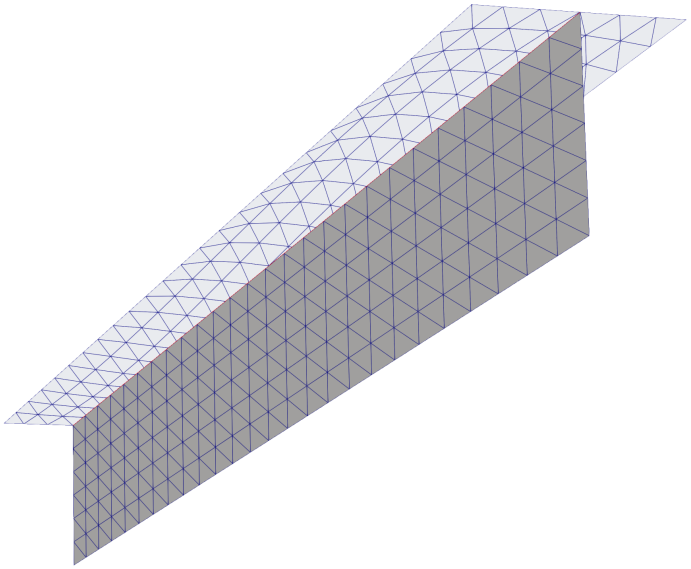
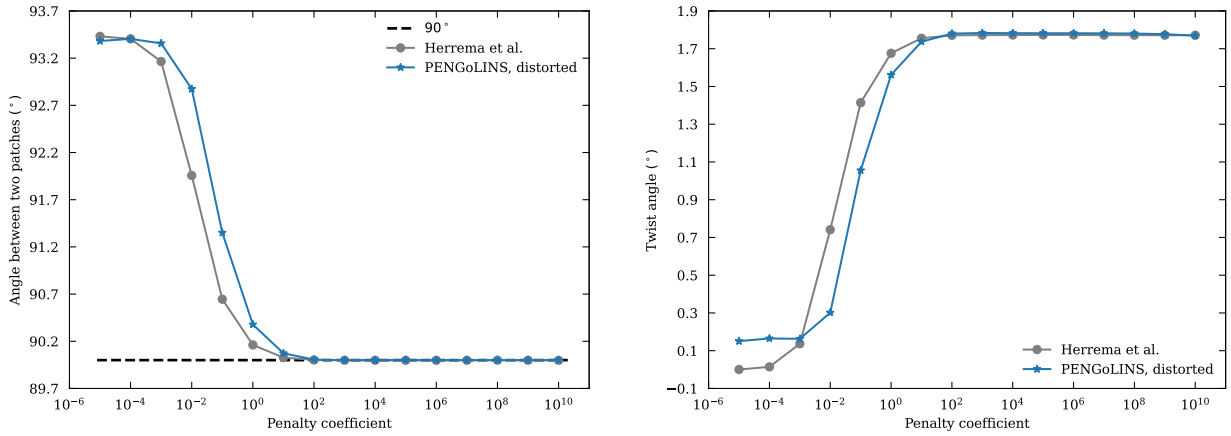


Figure 12: T-beam with a distorted parameterization of the top patch.

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 7 of this paper and [31]. We can conclude from the additional tests of this section that the



(a) (b)

Figure 13: 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 12. We again include the results of [31] for comparison, although the interpretation of α differs by a constant factor when using a different mesh.

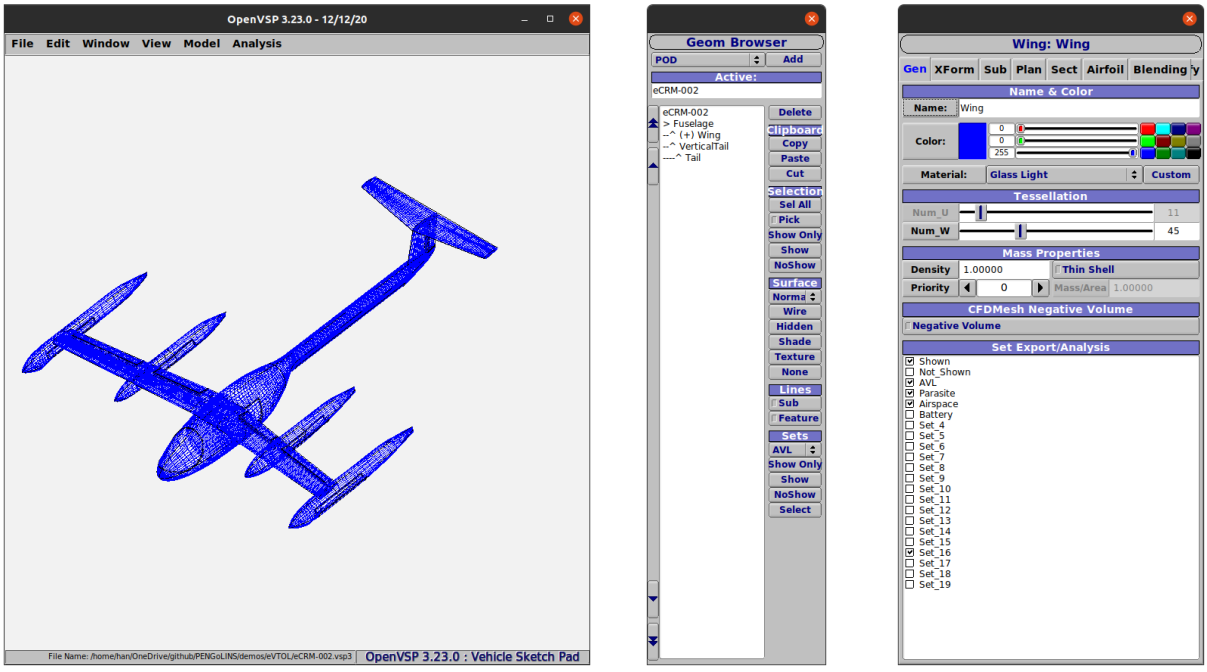
formulation (4) is robust with respect to different relative orientations of intersection curves and parametric coordinate lines.

5. Application to aircraft structures

Having verified the non-matching IGA formulation and its implementation in PENGoLINS, we now turn to our target application, namely, the analysis of aerospace structures. We first discuss the source of our upstream geometry in Section 5.1, then apply PENGoLINS to it in Section 5.2, thereby demonstrating a seamless design-through-analysis workflow that entirely bypasses the problematic mesh-generation stage.

5.1. Design of eVTOL wing geometry

The conceptual aircraft design tool OpenVSP 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 [63] 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 14a, alongside snapshots of the user interfaces for selecting geometry (Figure 14b) and adjusting design parameters for the wing (Figure 14c).

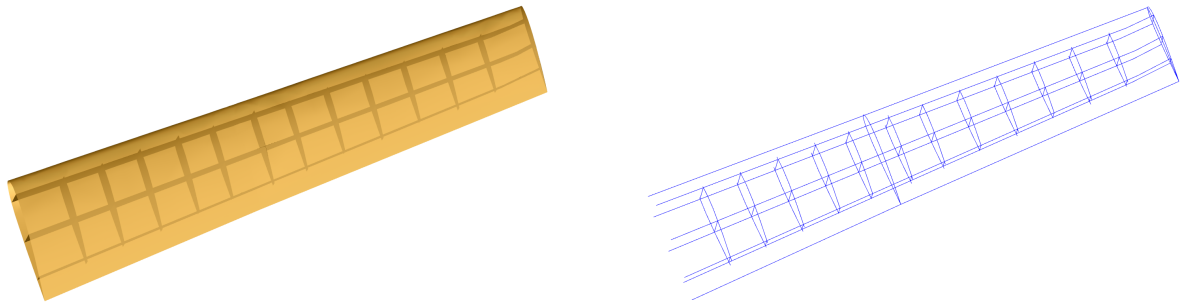


(a) (b) (c)

Figure 14: (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 industrial 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 3.1.1 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 3.1.1. 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 15.



(a) (b)

Figure 15: (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.

5.2. 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 5.1. Spline patches from this geometry are enriched via knot insertion for analysis, as shown in Figure 16.

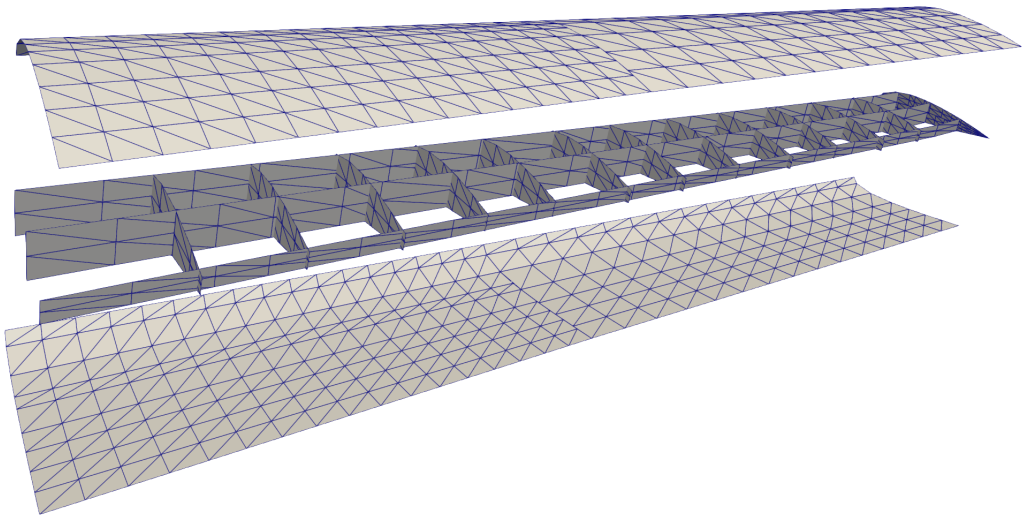


Figure 16: Exploded view of the eVTOL wing geometry of the eCRM-002 model, consisting of 21 NURBS shell patches with 87 intersections. The total number of displacement DoFs is 5524. Lower and upper surfaces are displaced vertically, to show the internal stiffeners.

The root of the wing is clamped in this analysis, and a distributed volumetric upward 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.[5] 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 $\mathrm{N/m^3}$. Figure 17 demonstrates the stress analysis result for the geometry in Figure 16 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 displacement at the wingtip on the trailing edge. This QoI is computed at several levels of refinement,

[5]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.

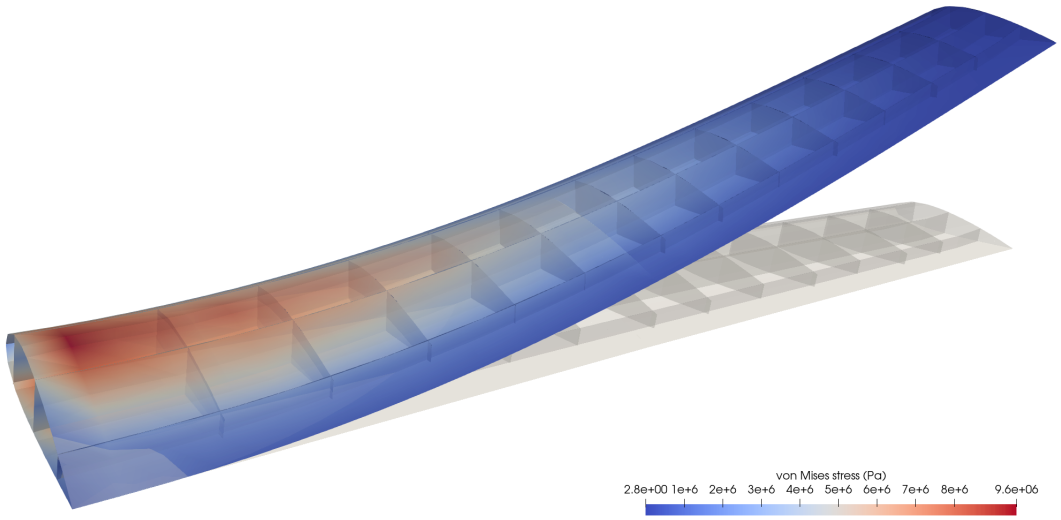


Figure 17: Distribution of von Mises stresses of the eVTOL wing. Wing displacements are scaled by a factor of 100, and the upper surfaces are translucent for visualization.

and the results are plotted in Figure 18. 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.

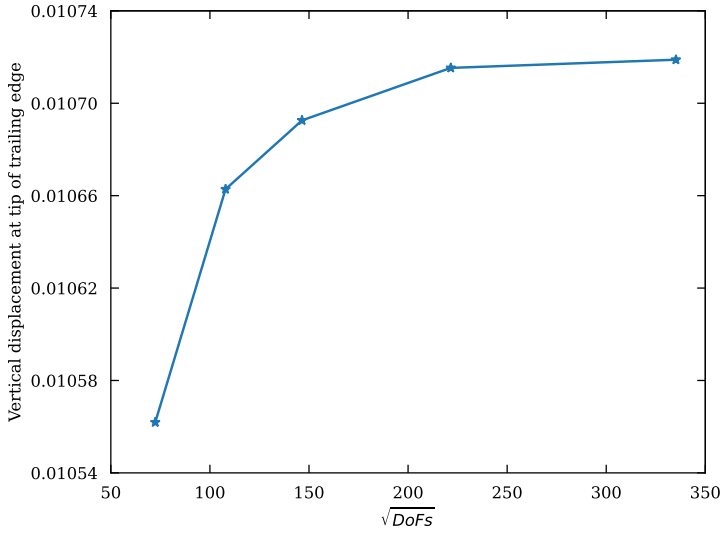
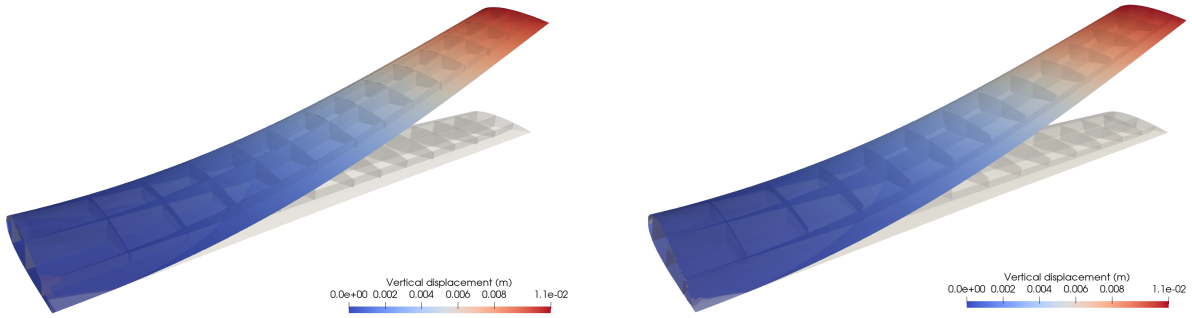


Figure 18: Convergence of the vertical displacement at the wingtip of the trailing edge.

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 [64] (i.e., triangles with quadratic Lagrange displacements and linear Crouzeix–Raviart rotations), similar to the didactic code example provided by [65]. The vertical displacement of the wingtip trailing edge in a converged finite element analysis with 1,262,709 DoFs is 0.010804 m.[6] 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 [61, 66], respectively). The displacement solutions from PENGoLINS and classical finite element analysis are compared in Figure 19, 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.



(a) (b)

Figure 19: (a) Displacement solution for an eVTOL wing using PENGoLINS. (b) Displacement solution using the Reissner–Mindlin shell element of [64].

6. Conclusion

In this paper, we have introduced a new open-source framework, PENGoLINS, which performs IGA on collections of shell structures with non-matching parameterizations, using a penalty-based coupling formulation similar to that of [31]. Our implementation leverages robust computational

[6] We do not claim that such a large number of DoFs is strictly necessary for accurate results with the formulation of [64], 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).

geometry operations from the Open Cascade 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.

Planned extensions of PENGoLINS include higher-fidelity constitutive modeling (e.g., of multi-layer composite laminates) and functionality to map source terms from external data (e.g., aerodynamic loads from computational fluid dynamics) onto models. Expansion of constitutive modeling in particular will be greatly streamlined by FEniCS's automation: direct access to computer algebra in the analysis code renders tedious and error-prone manual linearization of material models obsolete. Sophisticated data-driven constitutive modeling for Kirchhoff–Love shells has already been demonstrated using tIGAr [67], and the penalty-based coupling implemented here is largely independent of constitutive model.[7]

In the context of design optimization, our use of FEniCS's code generation provides efficient and automatic access to derivatives of the discrete problem's residual with respect to state and design variables, as needed by efficient minimization algorithms [68]. We plan to take advantage of this to perform multidisciplinary optimization (MDO) of eVTOL aircraft, using the OpenMDAO framework to incorporate PENGoLINS-based shell analysis as a modular component, alongside other models for batteries, motors, etc. We recently demonstrated the effectiveness of combining FEniCS and OpenMDAO in the context of multiphysics topology optimization [69], including an application to eVTOL battery packs [70].

Lastly, we note that PENGoLINS can be applied to a wide variety of applications beyond conceptual aircraft design. For example, both ShNAPr and the immersed fluid–structure interaction (FSI) library CouDALFISh [71] have been updated for compatibility with PENGoLINS, which has enabled exploratory FSI analysis of prosthetic heart valves using multi-patch parameterizations of leaflets, as shown in Figure 20. We anticipate that this software will be useful across multiple communities and welcome feedback, bug reports, and contributions through the public GitHub repository [40].

Acknowledgements

HZ was supported by start-up and fellowship funds provided through the University of California San Diego during the preparation of the original submission. HZ and RX were supported by NASA grant number 80NSSC21M0070 while preparing the revised submission.

[7]The only change required would be to identify some approximate scalar "stiffness" with units of pressure to replace the Young's modulus in the definitions of penalty parameters. The insensitivity to non-dimensional penalty coefficient demonstrated in Section 4 indicates that this stiffness must only be accurate to within about an order of magnitude.

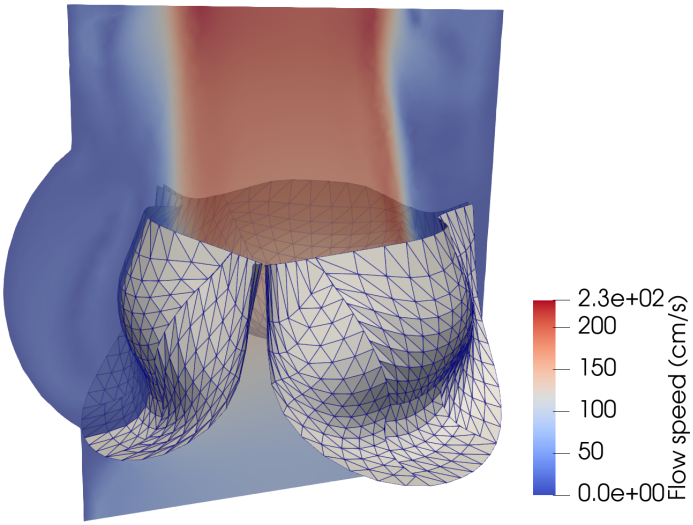


Figure 20: Preliminary FSI analysis of a prosthetic heart valve, using non-matching parameterizations of its leaflets.

References

[1] M. J. Duffy, S. R. Wakayama, and R. Hupp. A study in reducing the cost of vertical flight with electric propulsion. In *17th AIAA Aviation Technology, Integration, and Operations Conference*, page 3442, 2017.

[2] 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.

[3] A. Bacchini and E. Cestino. Electric VTOL configurations comparison. *Aerospace*, 6(3), 2019.

[4] 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.

[5] 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.

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

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

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

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

[35] 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.

[36] 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.

[37] 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.

[38] A. Logg, K.-A. Mardal, G. N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.

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

[40] https://github.com/hanzhao2020/PENGoLINS. PENGoLINS source code.

[41] A. Buganza Tepole, H. Kabaria, K.-U. Bletzinger, and E. Kuhl. Isogeometric Kirchhoff–Love shell formulations for biological membranes. *Computer Methods in Applied Mechanics and Engineering*, (0):–, 2015.

[42] 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.

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

[44] A. Logg and G. N. Wells. DOLFIN: Automated finite element computing. *ACM Trans. Math. Softw.*, 37(2):20:1–20:28, April 2010.

[45] 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.

[46] 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.

[47] 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.

[48] https://github.com/david-kamensky/ShNAPr. ShNAPr source code.

[49] 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.

[50] 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.

[51] 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.

[52] Thomas Paviot and J Feringa. Pythonocc. Technical report, 3D CAD/CAE/PLM development framework for the Python programming language, 2018.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] http://hangar.openvsp.org/. VSP hangar.

[64] 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.

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

[66] 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.

[67] W. Zhang, G. Rossini, D. Kamensky, T. Bui-Thanh, and M. S. Sacks. Isogeometric finite element-based simulation of the aortic heart valve: Integration of neural network structural material model and structural tensor fiber architecture representations. *International Journal for Numerical Methods in Biomedical Engineering*, 37(4):e3438, 2021.

[68] 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.

[69] 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*, 2021. In revision.

[70] J. Yan, M. Sperry, D. Kamensky, and J. T. Hwang. Multi-fidelity design optimization of battery packs for eVTOL aircraft. In *AIAA AVIATION 2021 FORUM*, 2021.

[71] https://github.com/david-kamensky/CouDALFISh. CouDALFISh source code.