Lehrstuhl für Computing in Civil and Building Engineering
TUM School of Engineering and Design
Technical University of Munich

Software Lab 2025

Next-gen Simulation: CAD-Integrated Isogeometric Analysis, Shell Structures, and Enhanced Visualization Frameworks

Matheus Ribeiro Vidal[1], Wataru Fukuda[2], Om Navale[3], Yousif Ali[4]

[1]matheus.ribeiro-vidal@tum.de
[2]wataru.fukuda@tum.de
[3]om.navale@tum.de
[4]yousif.ali@tum.de

April 17, 2026

Table of Contents

1 Matheus Ribeiro Vidal: Third Medium Contact – Material Modeling for Contact Problems

1.1 Introduction

Self-contacting geometries cannot be reliably computed in the IGAeasy framework, often leading to non-physical deformation behavior.

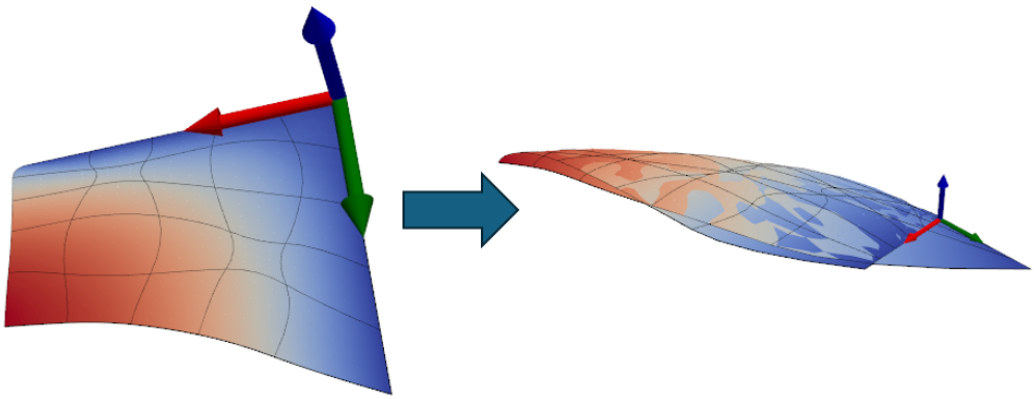


Figure 1: Non-physical deformation under excessive load

To address this limitation, a Third Medium Contact (TMC) material model is implemented to act as a virtual displacement limiter, enabling stable simulation of self-contact scenarios. The TMC approach avoids the computational complexity of traditional discrete contact detection algorithms by introducing an intermediate continuum, the "third medium", between potentially contacting surfaces [1]. This concept has been successfully extended to topology optimization problems with internal contact [2, 3]. The present implementation adopts a simplified formulation with explicit threshold-based stiffness activation, which effectively creates a compressible cushion layer that transmits load only during strong compression, with the intention of preserving smooth stress transitions between open and closed contact states.

1.2 Methodology

1.2.1 Strain Energy Function

The strain energy density of the third medium material is decomposed into isotropic and anisotropic contributions:

$$W = W_{\text{iso}} + W_{\text{aniso}} \tag{1}$$

The isotropic part follows a hyperelastic formulation expressed in terms of the invariants of the right Cauchy-Green deformation tensor $\mathbf{C} = \mathbf{F}^\top \mathbf{F}$:

$$W_{\text{iso}} = a_1 I_1 + a_2 I_2 + a_3 I_3 - d \ln \sqrt{I_3} \tag{2}$$

where $I_1 = \text{tr}(\mathbf{C})$, $I_2 = \frac{1}{2}\left(I_1^2 - \text{tr}(\mathbf{C}^2)\right)$, and $I_3 = \det(\mathbf{C})$ are the principal invariants. The parameters a_1, a_2, and a_3 are base isotropic coefficients controlling bulk and shear scaling. The coefficient $d = 2a_1 + 4a_2 + 2a_3$ ensures stress-free behavior in the reference configuration.

The anisotropic contribution provides directional stiffness for contact enforcement and is given by:

$$W_{\text{aniso}} = \frac{c_M}{a_4 + 1} \left(1 - J_4\right)^{a_4+1} \tag{3}$$

where $J_4 = \mathbf{n} \cdot \mathbf{C} \cdot \mathbf{n}$ is the pseudo-invariant associated with the preferred direction $\mathbf{n}$ (typically aligned with the contact normal), and a_4 is the anisotropic exponent controlling the sensitivity of the dynamic stiffness response.

1.2.2 Dynamic Stiffness Activation

A key feature of the TMC formulation is the deformation-triggered stiffness c_M, which activates only under compression. This is achieved through the following state-dependent function:

$$c_M = \begin{cases} 0 & \text{if } \det(\mathbf{F}) > \text{tol} \\ a_5 \left[\det(\mathbf{F})\right]^i & \text{if } \det(\mathbf{F}) \leq \text{tol} \end{cases} \tag{4}$$

where $\det(\mathbf{F})$ is the determinant of the deformation gradient tensor, a_5 is the base stiffness magnitude that scales the dynamic stiffness, i is the stiffness exponent controlling the $\det(\mathbf{F})$ dependence, and tol is a compression threshold acting as the activation cutoff. This formulation ensures that the third medium offers negligible resistance when bodies separate (tension/expansion) but provides rapidly increasing resistance as they compress beyond the threshold (contact). The material parameters are summarized in Table 1.

Table 1: Third Medium Contact material parameters.

Parameter	Meaning	Role
a_1, a_2, a_3	Base isotropic coefficients	Bulk and shear scaling
a_4	Anisotropic exponent	Controls dynamic stiffness sensitivity
a_5	Base stiffness magnitude	Scales dynamic stiffness
i	Stiffness exponent	Controls $\det(\mathbf{F})$ dependence
tol	Compression threshold	Activation cutoff

1.2.3 Second Piola-Kirchhoff Stress

The second Piola-Kirchhoff stress tensor is obtained from the strain energy via the standard hyperelastic relation:

$$\mathbf{S} = 2\frac{\partial W}{\partial \mathbf{C}} \tag{5}$$

For the isotropic contribution:

$$\frac{\partial W_{\text{iso}}}{\partial \mathbf{C}} = a_1 \mathbf{I} + a_2 \left(I_1 \mathbf{I} - \mathbf{C}\right) + a_3 I_3 \mathbf{C}^{-1} - \frac{d}{2I_3} \mathbf{C}^{-1} \tag{6}$$

The anisotropic contribution adds:

$$\frac{\partial W_{\text{aniso}}}{\partial \mathbf{C}} = c_M \left(1 - J_4\right)^{a_4} \left(\mathbf{n} \otimes \mathbf{n}\right) \tag{7}$$

1.2.4 Material Tangent Stiffness

The material tangent tensor, required for Newton-Raphson iteration in the nonlinear solver, is computed as:

$$\mathbb{C} = 2\frac{\partial \mathbf{S}}{\partial \mathbf{C}} = 4\frac{\partial^2 W}{\partial \mathbf{C} \partial \mathbf{C}} \tag{8}$$

The isotropic part follows a Neo-Hookean-like structure:

$$\mathbb{C}^{\text{iso}}_{ijkl} = \lambda\, C^{-1}_{ij} C^{-1}_{kl} + \mu \left[\frac{1}{2}\left(C^{-1}_{ik} C^{-1}_{jl} + C^{-1}_{il} C^{-1}_{jk} \right) - \frac{1}{3} C^{-1}_{ij} C^{-1}_{kl} \right] \tag{9}$$

where $\lambda = K - \frac{2}{3}\mu$ with $K = a_1$ (bulk-like parameter) and $\mu = a_2$ (shear-like parameter). The anisotropic contribution to the tangent is:

$$\mathbb{C}^{\text{aniso}}_{ijkl} = c_M \left(1 - J_4\right)^{a_4} N_{ij} N_{kl} \tag{10}$$

where $\mathbf{N} = \mathbf{n} \otimes \mathbf{n}$ is the structural tensor.

1.2.5 Voigt Notation

For computational efficiency in the finite element assembly, the fourth-order tangent tensor is stored in Voigt notation. In 2D (plane strain/stress), the mapping uses the convention:

$$\{11, 22, 12\} \to \{1, 2, 3\} \tag{11}$$

yielding a 3×3 matrix. In 3D:

$$\{11, 22, 33, 23, 13, 12\} \to \{1, 2, 3, 4, 5, 6\} \tag{12}$$

yielding a 6×6 matrix. Appropriate scaling factors of $\frac{1}{2}$ are applied to shear components during the conversion to maintain consistency with engineering strain definitions.

1.3 Results

1.3.1 Class Implementation

The material model is implemented as a Python class `ThirdContactMaterial` within the IGAeasy framework, as a material model script. The class is instantiated with the material parameters as follows:

```
material = ThirdContactMaterial(a1, a2, a3, a4, a5,
                                stiffness_exponent, tol=0.01)
```

where `a1`, `a2`, `a3` are the base isotropic coefficients, `a4` is the anisotropic exponent, `a5` is the base stiffness magnitude, `stiffness_exponent` (denoted i in the formulation) controls the $\det(\mathbf{F})$ dependence, and `tol` is the compression threshold with a default value of 0.01.
The class provides the following methods for use in the finite element assembly:

- `strain_energy(F, n=None)`:

  Returns the strain energy density W for a given deformation gradient $\mathbf{F}$ and optional fiber direction $\mathbf{n}$.

- `compute_stress_2D(F, n=None)` and `compute_stress_3D(F, n=None)`:

  Return the second Piola-Kirchhoff stress tensor $\mathbf{S}$ for 2D and 3D problems, respectively.

- `get_C_voigt_cart_2D(F, n=None)` and `get_C_voigt_cart_3D(F, n=None)`:

  Return the material tangent stiffness in Voigt notation as a 3×3 matrix (2D) or 6×6 matrix (3D), suitable for direct use in element stiffness assembly.

If the fiber direction `n` is not provided, the material behaves isotropically.

1.3.2 Material Behavior Verification

To verify the correct implementation of the dynamic stiffness activation, a parameter sweep was conducted over a range of $\det(\mathbf{F})$ values. Figure 2 shows the resulting maximum stress and dynamic stiffness c_M as functions of $\det(\mathbf{F})$. The results confirm the expected behavior: when $\det(\mathbf{F}) > \text{tol}$ (material in tension/expansion), the dynamic stiffness remains zero and the material offers negligible resistance. As $\det(\mathbf{F})$ decreases below the threshold, indicating compression, the stiffness activates and increases rapidly following the prescribed power law $c_M = a_5[\det(\mathbf{F})]^i$. This demonstrates that the implementation effectively creates a compressible cushion layer that transmits load only during strong compression, which is ideal for gaps or porous interfaces in contact problems.

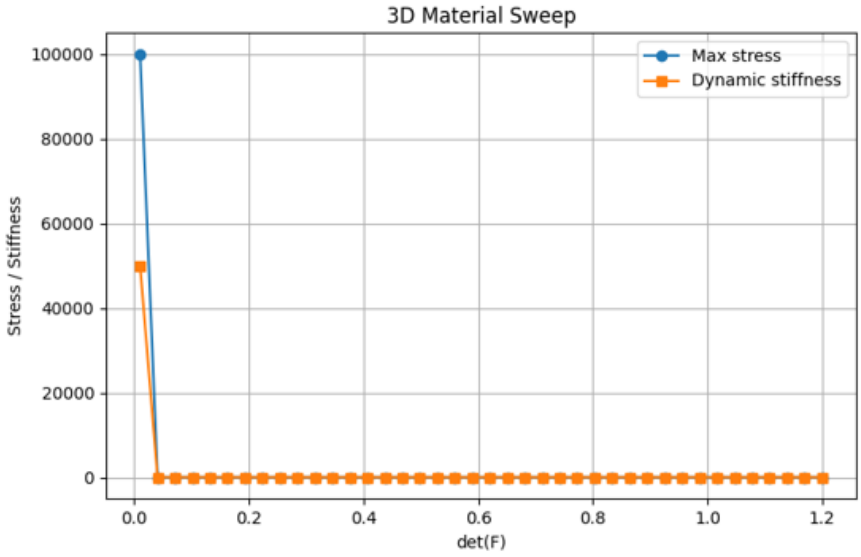


Figure 2: 3D material sweep showing maximum stress and dynamic stiffness as functions of $\det(\mathbf{F})$. The stiffness activates only when $\det(\mathbf{F}) \leq \text{tol}$, confirming the compression-triggered behavior.

1.3.3 Example Case: 2D Membrane Under Compression

To demonstrate the practical application of the Third Medium Contact material, a 2D membrane example case was set up. The geometry consists of a rectangular domain

filled with the third medium material, constrained at one edge with Dirichlet boundary conditions (fixed edge) and subjected to a prescribed line load on the opposite edge to induce compression.

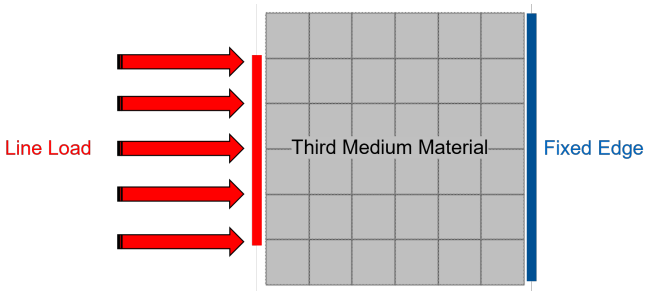


Figure 3: Example case setup

The simulation was performed using the nonlinear solver already implemented in IGAeasy with load stepping to handle the material nonlinearity. Figure 4 shows the displacement magnitude at three successive load steps. The results demonstrate the expected large deformations in the third medium material before the stiffness activation threshold is reached. The material successfully deforms under the applied load, exhibiting the characteristic behavior of a soft, compressible layer.

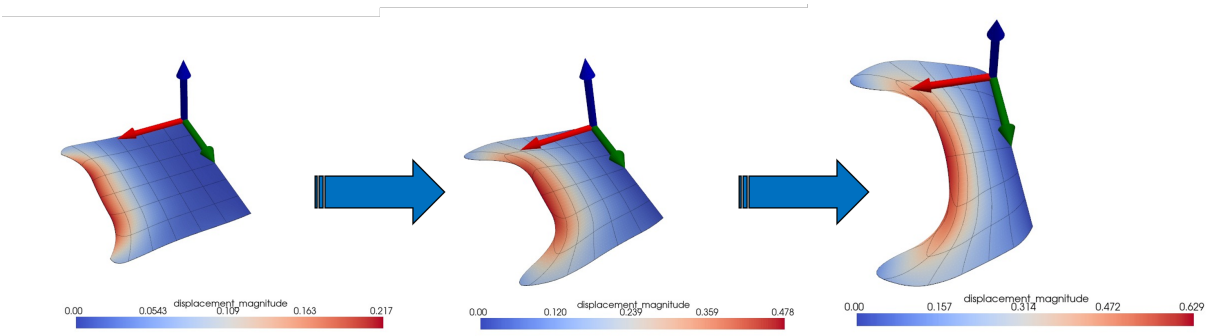


Figure 4: Deformation sequence of the 2D membrane under increasing load, showing displacement magnitude.

1.3.4 Convergence Challenges

While the implementation correctly captures the intended material behavior, convergence difficulties were encountered in the nonlinear solver when using the Third Medium Contact material. The main hypothesis is that these challenges arise primarily due to the sudden jump in stiffness at the activation threshold $\det(\mathbf{F}) = \mathrm{tol}$. The discontinuous nature of the stiffness function leads to ill-conditioning of the tangent stiffness matrix near the transition point, causing the Newton-Raphson iterations to struggle or fail to converge.
With load stepping, partial results were obtained showing the expected large deformations before the stiffness jump. However, achieving full convergence through the contact activation phase remains challenging with the current nonlinear solver implementation in IGAeasy.

1.3.5 Limitations

The current implementation also has some limitations in regards to modeling real world contact physics that should be noted. The model enforces contact in the normal direction through the compression-activated stiffness, but lacks tangential contact behavior. Therefore, it does not accurately represent friction between contacting surfaces. For applications where frictional effects are important, additional model extensions would be required.

1.4 Conclusion

The Third Medium Contact material model was successfully implemented within the IGAeasy framework as a Python class. The implementation provides a complete hyperelastic formulation with deformation-triggered stiffness activation, including strain energy evaluation, second Piola-Kirchhoff stress computation, and material tangent stiffness in Voigt notation for both 2D and 3D problems.

Material behavior verification through parameter sweeps confirmed that the dynamic stiffness activates correctly under compression, following the prescribed power law when $\det(\mathbf{F}) \leq \mathrm{tol}$ and remaining inactive otherwise. The example case demonstrated that the material effectively creates a compressible cushion layer capable of undergoing large deformations before stiffness activation.

However, convergence difficulties were encountered in the nonlinear solver due to the sudden jump in stiffness at the activation threshold. With load stepping, partial results were obtained showing the expected behavior before the stiffness jump, but achieving full convergence through the contact activation phase remains challenging with the current solver implementation.

Despite these challenges, the implementation provides a foundation for contact modeling in IGAeasy that avoids discrete contact detection algorithms while preserving smooth stress transitions between open and closed contact states.

1.5 Future Work

Some improvements and extensions identified for future development were:

- *Nonlinear solver improvements:* Reworking the nonlinear solver to better account for the sudden jump in stiffness. This could involve implementing adaptive load stepping with smaller increments near the activation points.
- *Tangential contact behavior:* The current implementation enforces contact only in the normal direction through compression-activated stiffness, and therefore does not accurately represent friction. Extending the model to include tangential stiffness components would enable proper frictional contact simulation.
- *Parameter sensitivity study:* A systematic study of the material parameters (a_1-a_5, i, tol) and their influence on convergence behavior and contact accuracy would provide guidelines for practical application of the model.

2 Wataru Fukuda: Post-processing – Visualization of exact IGA geometry with VTK and ParaView

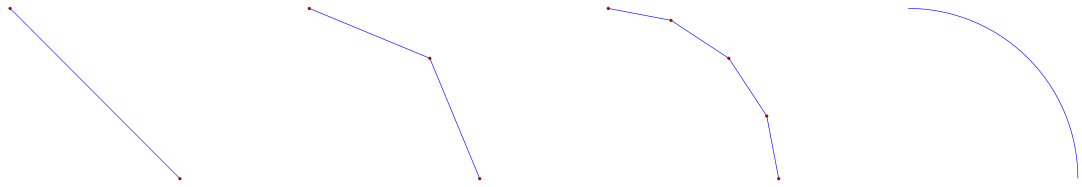
2.1 Introduction

Isogeometric Analysis (IGA) was developed to bridge the gap between the created smooth geometry and meshing by linearization by using the exact basis functions from CAD, such as NURBS, directly within the analysis solver. However, a drawback remains in the visualization stage, where these high-fidelity IGA results are typically forced back into a linearized format. This reliance on linearization for post-processing creates a bottleneck that masks the true accuracy of IGA. This part aims to resolve this inconsistency by developing a visualization framework that maintains the exact geometric integrity of IGA throughout the entire pipeline, from initial calculation to the final visual output.

2.2 Methodology

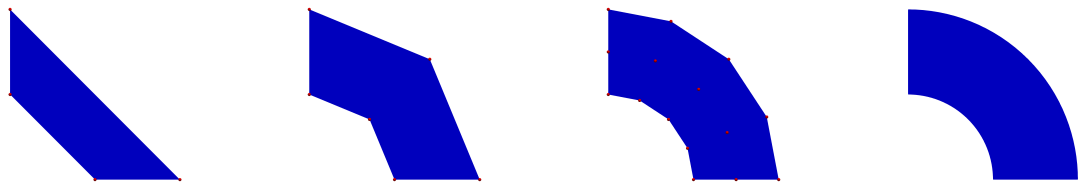
2.2.1 VTK

To address the problem described in the previous section, we employ the Visualization Toolkit (VTK) to represent the exact IGA geometry. According to the paper published by Kitware (link to page), the developer of ParaView, new element types capable of representing exact Bézier-element geometry have been implemented in VTK, and these formats are fully readable in ParaView. Examples of these newly introduced element types are shown below:



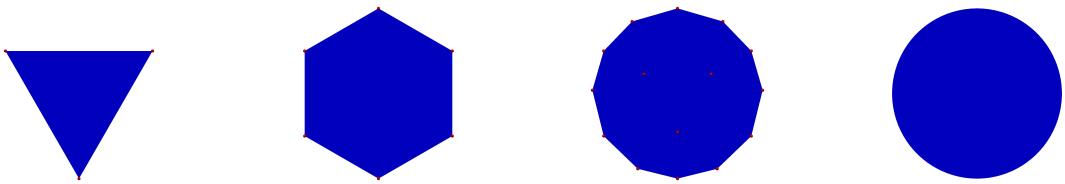
(a) div 0 (b) div 1 (c) div 2 (d) div 5

Figure 5: BEZIER_CURVE at different subdivision levels



(a) div 0 (b) div 1 (c) div 2 (d) div 5

Figure 6: BEZIER_QUADRILATERAL at different subdivision levels



(a) div 0 (b) div 1 (c) div 2 (d) div 5

Figure 7: BEZIER_TRIANGLE at different subdivision levels

In our project, the BEZIER_QUADRILATERAL is only employed to represent 2D IGA geometry. It should be emphasized that each of these represents a single Bézier element, not multiple B-spline elements. Therefore, additional processing is required to correctly represent the multiple B-spline elements typically used in our project.

2.2.2 Bezier Extraction Operator

To construct a set of Bézier elements from multiple B-spline elements, the procedure known as knot insertion is required. Applying this technique reduces the continuity of the basis functions at element boundaries without changing the underlying geometry. In other words, the geometric representation remains unchanged, while the functional continuity is reduced to the desired level. After this preprocessing step, the original set of B-spline elements can be successfully converted into an equivalent set of Bézier elements.
The meaning of knot insertion is adding a new knot to the existing knot vector without changing the shape of the curve. The number of basis functions and control points increases by one. The transformation is defined as follows.
To the existing knot vector:

$$\Xi = \{\Xi_1, \Xi_2, \ldots, \Xi_n\} \tag{13}$$

a new knot $\Xi_k \leq \Xi \leq \Xi_{k+1}$ is inserted, and a new knot vector:

$$\Xi = \{\Xi_1, \Xi_2, \ldots, \Xi_k, \Xi, \Xi_{k+1}, \ldots, \Xi_n\} \tag{14}$$

is obtained. The basis functions of the patch before knot insertion, N_A, are expressed in terms of the basis functions after insertion, $\overline{N}_A$, and a transformation operator K_{AB}:

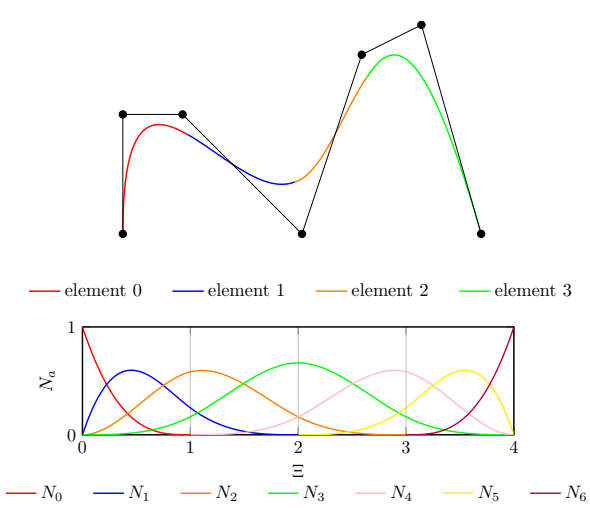
$$N_A = \mathrm{K}_{AB}\overline{N}_B \tag{15}$$

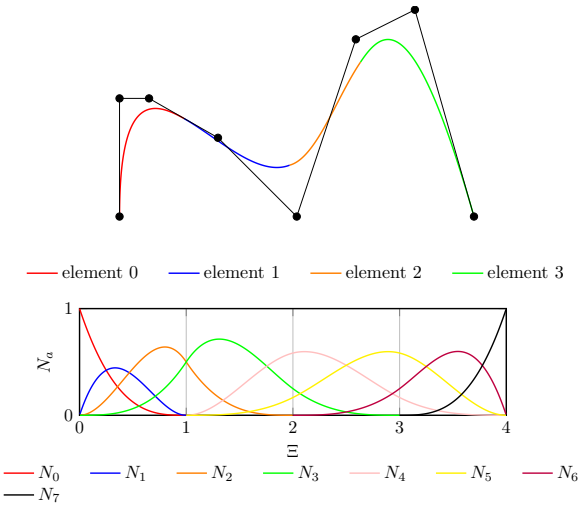
The transformation operator is defined as a $nn \times (nn+1)$ matrix:

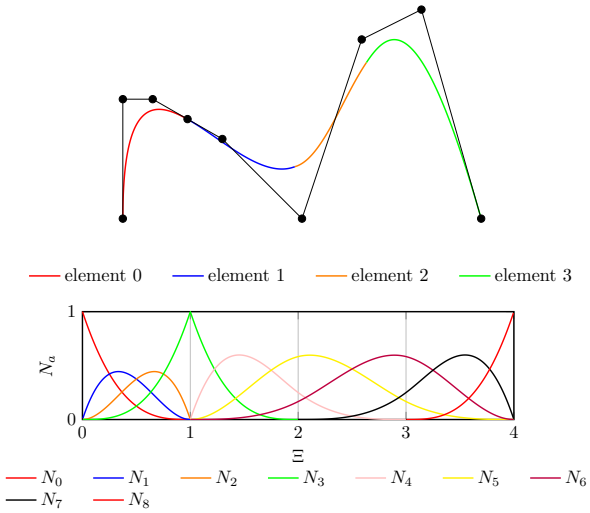
$$\left[\mathbf{K}_{AB}\right] = \begin{bmatrix} \alpha_1 & 1-\alpha_2 & 0 & \ldots & & & 0 \\ 0 & \alpha_2 & 1-\alpha_3 & 0 & \ldots & & 0 \\ 0 & 0 & \alpha_3 & 1-\alpha_4 & 0 & \ldots & 0 \\ \vdots & & & & & & \\ 0 & \ldots & & & 0 & \alpha_{nn} & 1-\alpha_{nn+1} \end{bmatrix} \quad \alpha_a = \begin{cases} 1 & 1 \leq a \leq k-p \\ \frac{\Xi-\Xi_a}{\Xi_{a+p}-\Xi_q} & k-p+1 \leq a \leq k \\ 0 & k+1 \leq \Xi_a \leq nn+1 \end{cases}$$

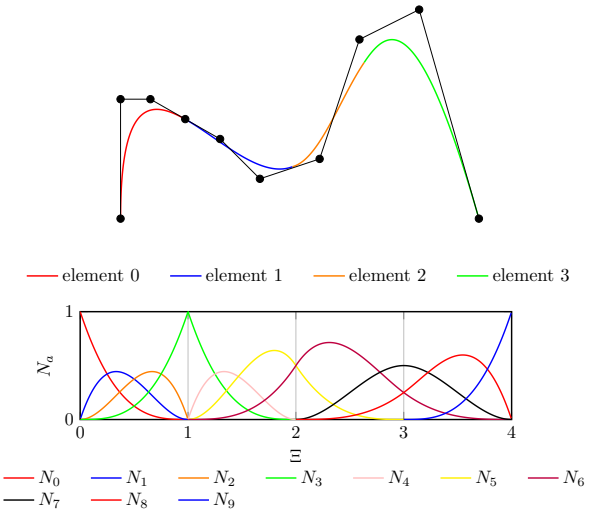
The following figures (Fig. 11) illustrate the knot insertion process for a problem setting with polynomial order 3. In this example, the initial number of control points is seven. This knot insertion is described in the following form;

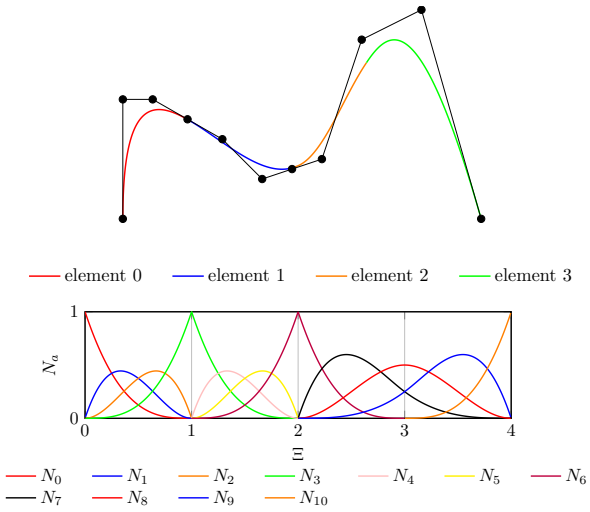
$$N_A = \underbrace{\mathrm{K}^6_{AB}\mathrm{K}^5_{BC}\mathrm{K}^4_{CD}\mathrm{K}^3_{DE}\mathrm{K}^2_{EF}\mathrm{K}^1_{FG}}_{\mathbf{C}^{\text{total}}}\overline{N}_G \tag{16}$$

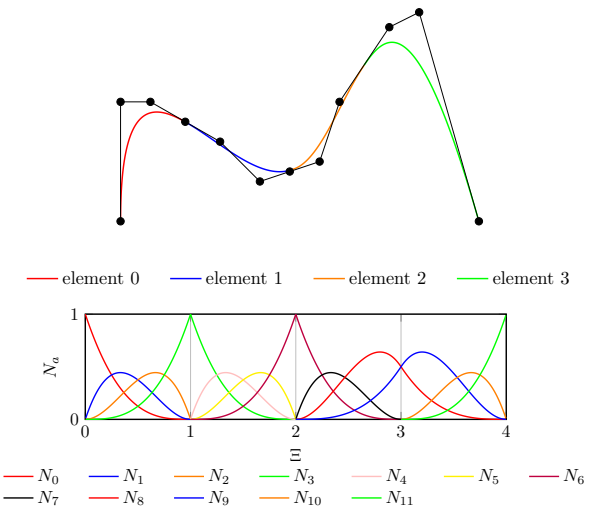

element 0
element 1
element 2
element 3
N_a
1
0
0
1
2
3
4
Ξ
N_0
N_1
N_2
N_3
N_4
N_5
N_6
N_7


element 0
element 1
element 2
element 3
N_a
1
0
0
1
2
3
4
Ξ
N_0
N_1
N_2
N_3
N_4
N_5
N_6
N_7


element 0
element 1
element 2
element 3
N_a
1
0
0
1
2
3
4
Ξ
N_0
N_1
N_2
N_3
N_4
N_5
N_6
N_7
N_8


element 0
element 1
element 2
element 3
N_a
1
0
0
1
2
3
4
Ξ
N_0
N_1
N_2
N_3
N_4
N_5
N_6
N_7
N_8
N_9


element 0
element 1
element 2
element 3
N_a
1
0
0
1
2
3
4
Ξ
N_0
N_1
N_2
N_3
N_4
N_5
N_6
N_7
N_8
N_9
N_{10}


element 0
element 1
element 2
element 3
N_a
1
0
0
1
2
3
4
Ξ
N_0
N_1
N_2
N_3
N_4
N_5
N_6
N_7
N_8
N_9
N_{10}
N_{11}

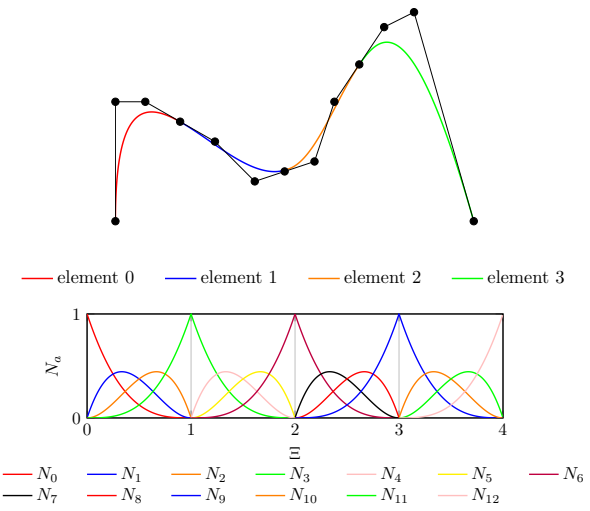
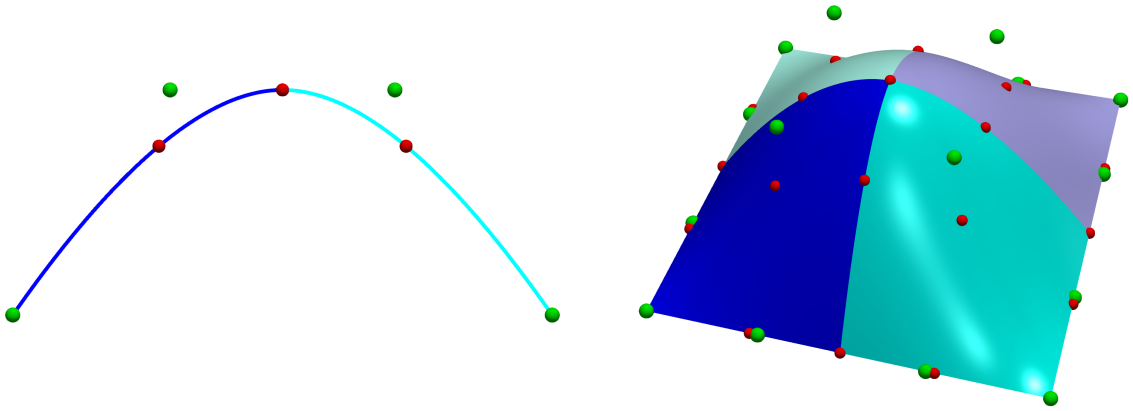


Figure 11: knot insertion

The size of the operator $\mathbf{C}^{\text{total}}$ is $nn \times (nn + 6)$, where n is the number of nodes before knot insertion. This indicates that inserting knots 6 times increases the number of control points by 6, until the FEM patch is obtained.

2.2.3 Test of VTK using Bezier Extraction Operator

After performing knot insertion and obtaining a set of Bezier elements, a test configuration of VTK was done for a simple 1D and 2D case(Fig. 12).



(a) 1D example (b) 2D example

Figure 12: VTK configuration test (green: initial control points, red: Bezier control points)

The newly introduced element type BEZIER_QUADRILATERAL is employed in this example. The varying shades of blue indicate individual Bézier elements. By extending

this basic usage of VTK to the context of our full project, we can obtain an exact geometric representation of the IGA model, as demonstrated in the following section.

2.3 Result

2.3.1 Displacement Field

Fig. 13–16 show the result for the geometry, halfpipe.

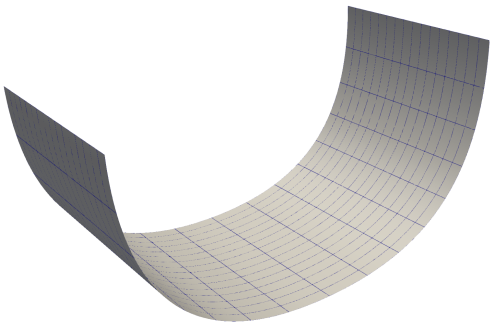


Figure 13: halfpipe without refinement

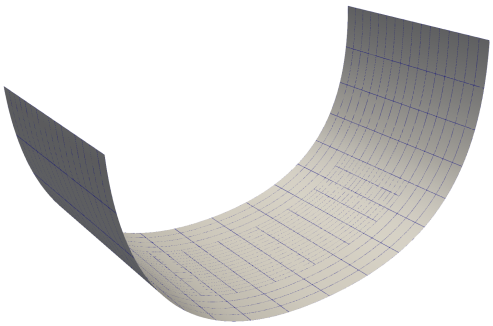


Figure 14: halfpipe with refinement

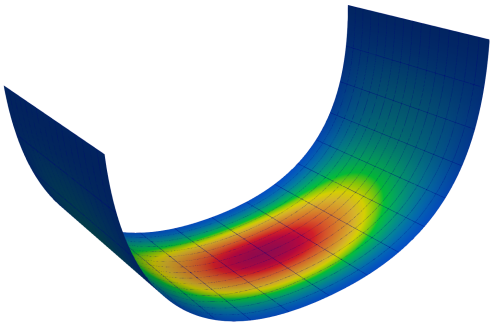


Figure 15: displacement for halfpipe without refinement

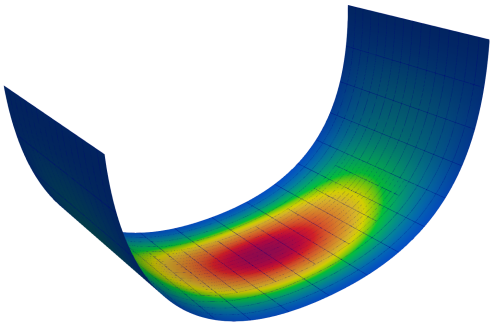


Figure 16: displacement for halfpipe with refinement

Fig. 17–20 show the result for the geometry, roof.

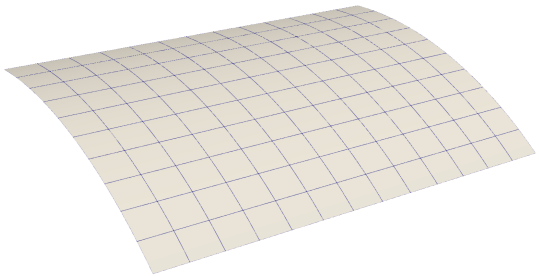


Figure 17: roof without refinement

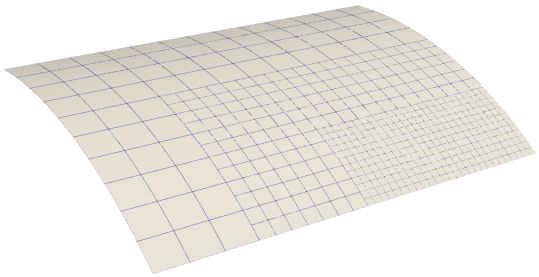


Figure 18: roof with refinement

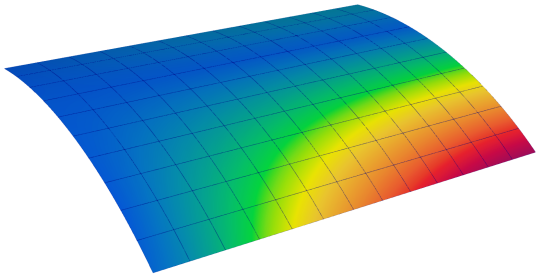


Figure 19: displacement for roof without refinement

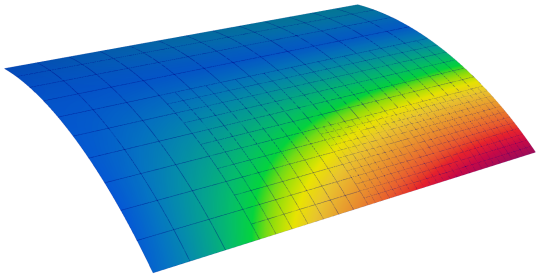


Figure 20: displacement for roof with refinement

Fig. 21–24 show the result for the geometry, roof, with trimming on.

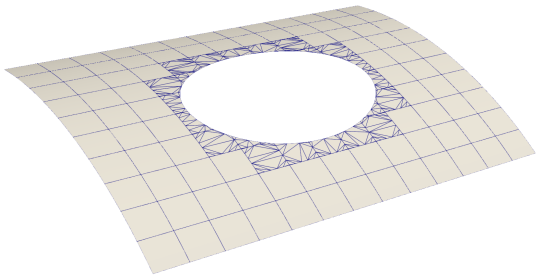


Figure 21: trimming

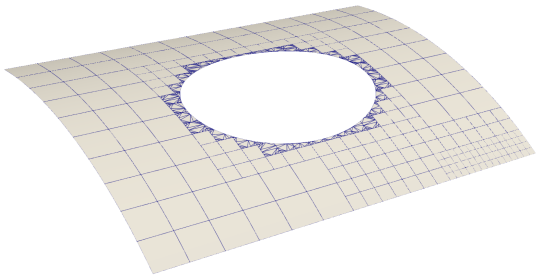


Figure 22: trimming and refinement

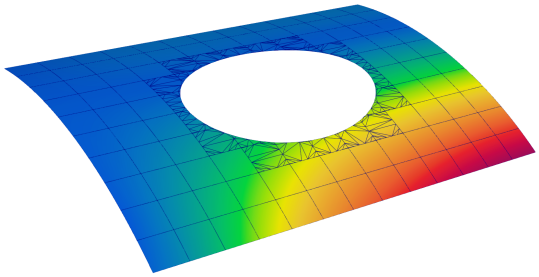


Figure 23: displacement with trimming

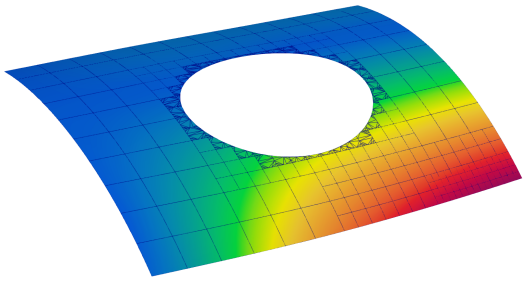


Figure 24: displacement with trimming and refinement

Especially for the combination of trimming and refinement, the region that still requires triangulation decreases as refinement is applied around it. This proved that this conceptually most challenging region can be perfectly described within this approach.

2.3.2 Control Points

Additional functionality that I implemented in this project is illustrated below. This allows us to export the control points for each element, which can be used for debugging and confirming the influence of each control point.

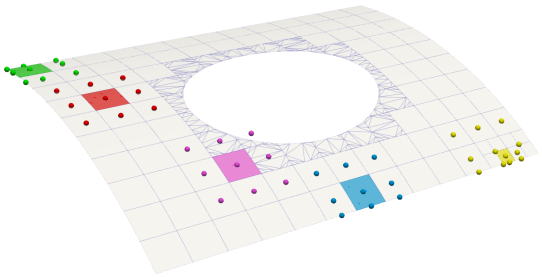


Figure 25: control points for each element

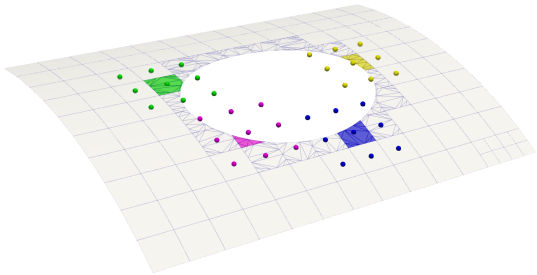


Figure 26: control points for each element with trimming

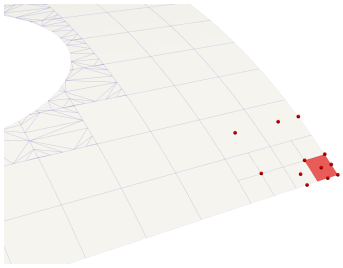


Figure 27: ex1

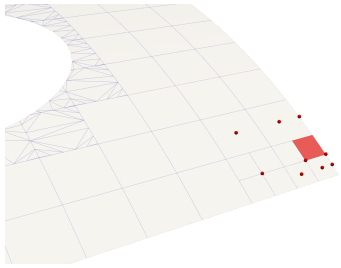


Figure 28: ex2

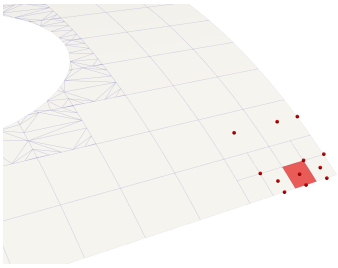


Figure 29: ex3

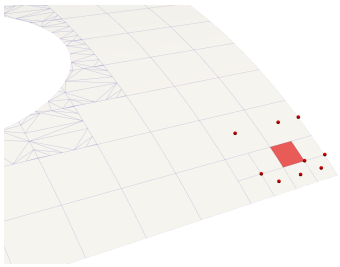


Figure 30: ex4

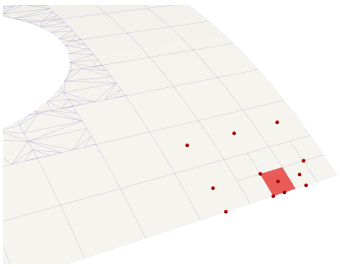


Figure 31: ex5

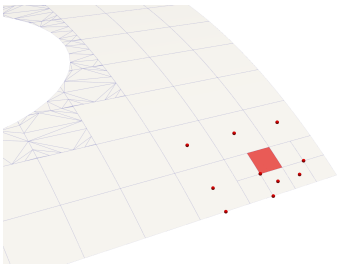


Figure 32: ex6

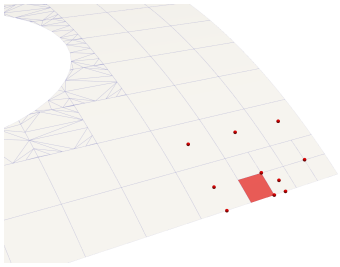


Figure 33: ex7

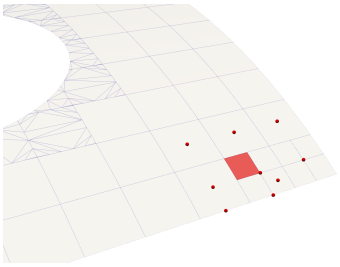


Figure 34: ex8

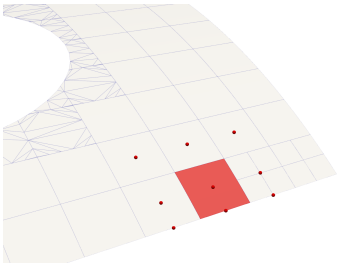


Figure 35: ex9

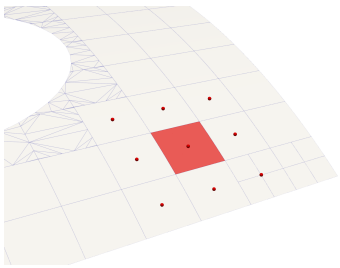


Figure 36: ex10

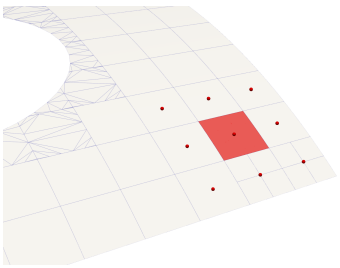


Figure 37: ex11

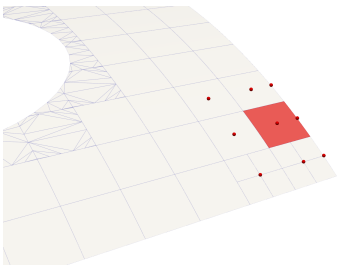


Figure 38: ex12

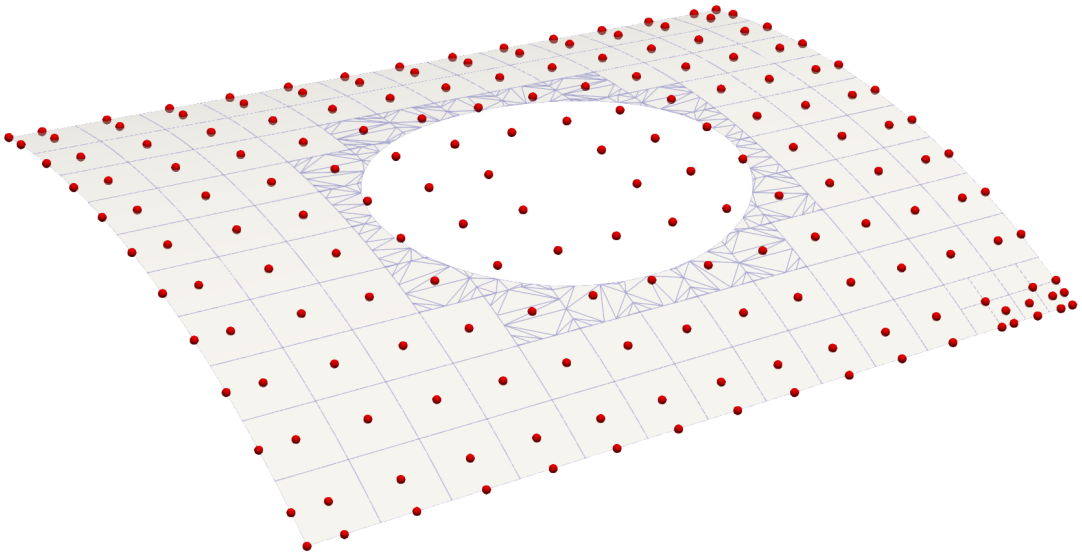


Figure 39: all active control points

2.4 Conclusion

In this project, the exact geometry visualization and exact displacement distribution on it were achieved. Even more, it supports all kinds of elements with refinement, trimming, and both of them. For debugging purposes, an output method that investigates all associated control points for each element is also implemented.

2.5 Future Work

Regarding the trimmed elements, triangulation and linearization are still employed to visualize the IGA element. This is the limitation in the current workflow. One approach to minimize this drawback is to refine the near region of trimming as shown in Fig. 22. For further improvement on this issue, a new approach needs to be developed.

3 Om Navale, Yousif Ali: Stress Smoothening and Visualization

3.1 Introduction

After numerical analysis is completed, stresses are obtained from the primary displacement solution and are therefore more sensitive to numerical inaccuracies than the displacement field itself. Within the IGAeasy Python package, direct element-wise visualization of these stresses often produces non-smooth and discontinuous stress patterns that are difficult to interpret, with clear jumps across adjacent elements.

To enable meaningful visualization while maintaining reasonable computational cost, stress-smoothing and visualization techniques are introduced. These methods are inspired by classical concepts from the finite element method (FEM) literature and are adapted here to the isogeometric analysis (IGA) framework.

The necessity of this approach is illustrated using a benchmark pinched-cylinder problem solved in IGAeasy, where directly plotted stresses reveal pronounced element-wise discontinuities. This observation motivates the proposed stress-smoothing strategy, whose effectiveness is demonstrated in the results section through the reconstruction of continuous and interpretable stress fields.

Length **L**	600.0
Radius **R**	300
Thickness **t**	3
Young's modulus **E**	3.00E+06
Poisson's ratio **v**	0.3
Boundary conditions	Constrained at each end by rigid diaphragm, $u_x = u_y = \phi_z = 0$
Loading	Opposing radial loads P=1.0

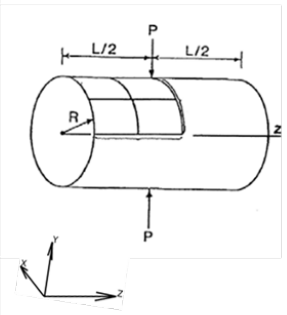


Figure 40: Problem setup pinched cylinder benchmark [4]

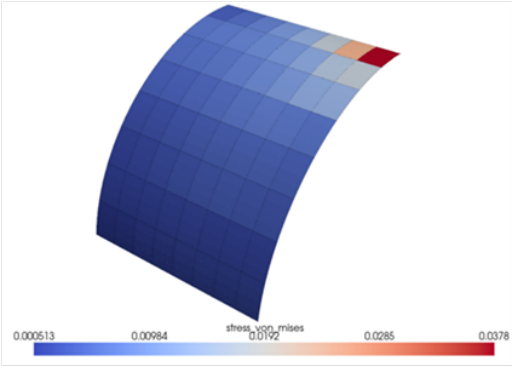


Figure 41: Discontinuous stress plot of pinched cylinder benchmark

3.2 Methodology

3.2.1 Stress interpolation

After stresses are computed within each element, they need to be transferred to the element corner points for visualization and post-processing. Two general strategies are commonly employed: direct evaluation of stresses at the element corners using parametric coordinates, or extrapolation from Gauss integration point values. The second approach is adopted here due to the higher accuracy typically achieved at Gauss points [5].

During analysis, stresses are evaluated at the in-plane Gauss points during the element routine, where individual stress components are accessed and the corresponding von Mises stress values are computed and stored together with their parametric coordinates. In addition, the coordinates of each element corner are recorded. These Gauss-point stress values and their associated coordinates, along with the element corner locations, are then passed to a bilinear dependency function, which approximates the stresses at the element corner points using a least-squares method, which when visualized, smooth stress variation within each element can be seen.

For the extrapolation from Gauss points to element corners, two alternatives were considered: direct polynomial interpolation and a least squares method (LSM).Standard interpolation requires a polynomial degree of N-1, where N is the number of sampling points, which can lead to oscillatory behaviour and increased computational cost. In contrast, the least squares approach employs a lower order polynomial, avoids oscillations, and remains computationally efficient. Based on these considerations, a least squares formulation using numpy.linalg.lstsq was selected.

3.2.2 Stress Visualization

Stress values evaluated at shared element corners generally differ between neighboring elements, as stress continuity is not enforced across element boundaries. As a result, the stress field exhibits C-1 continuity between elements while remaining smooth locally within each element. Therefore, stress averaging is required to construct a continuous and interpretable stress field for visualization.

As an initial approach, a simple averaging procedure was implemented in which stress values at each element corner were averaged by assigning equal weight to all contributing elements. While this method produces a continuous stress field across elements, it assumes uniform element size and therefore may lead to inaccurate approximations when elements differ in physical area. To overcome this limitation, a weighted averaging scheme was introduced, where each element's contribution is scaled by its physical area. This ensures that larger elements contribute proportionally more to the corner stress value, resulting in a more accurate stress representation. The parametric element area was not used, since it does not reflect geometric deformation in the physical domain.

The weighted corner stress was computed as

$$\sigma_{\text{corner}} = \frac{\sum (\sigma_e \, A_e)}{\sum A_e}$$

where σ_e is the corner stress contribution from element e and A_e is its physical area. The physical area of each element was evaluated using Gaussian quadrature:

$$A_e = \sum_{i=1}^{n_{gp}} w_i \, \det(J_i)$$

where w_i denotes the Gauss integration weight and $det(J_i)$ is the determinant of the Jacobian matrix at the i^{th} integration point and n_{gp} correspond to number of gauss point in the element.

These computed element areas were subsequently employed as weighting factors in

the stress averaging process, leading to a smooth and continuous stress visualization with improved physical accuracy compared to simple averaging.

3.2.3 Conceptual example illustration

To provide a clear visual intuition of the stress-smoothing procedures described above, a simplified one-dimensional IGA conceptual example is presented. The term conceptual indicates that the numerical values used are purely illustrative and do not represent physically meaningful quantities but are intended to demonstrate how the smoothing methodology is implemented within the IGAeasy framework.

Consider a open knot vector of a 1 D element:

k.v. = [0, 0, 0, 0.25, 0.75, 1, 1, 1]

Figure 42: 1D IGA element based on open knot vector

The Gauss-point stress values for each element are marked and plotted, stress values are denoted as $\sigma_{E_n}^{GP_i}$, where n represents the element index and i denotes the corresponding Gauss integration point, in Figure 43.Further, in Figure 44 the Gauss-point stresses are fitted using a least-squares method to obtain approximated stress values at the element corners.

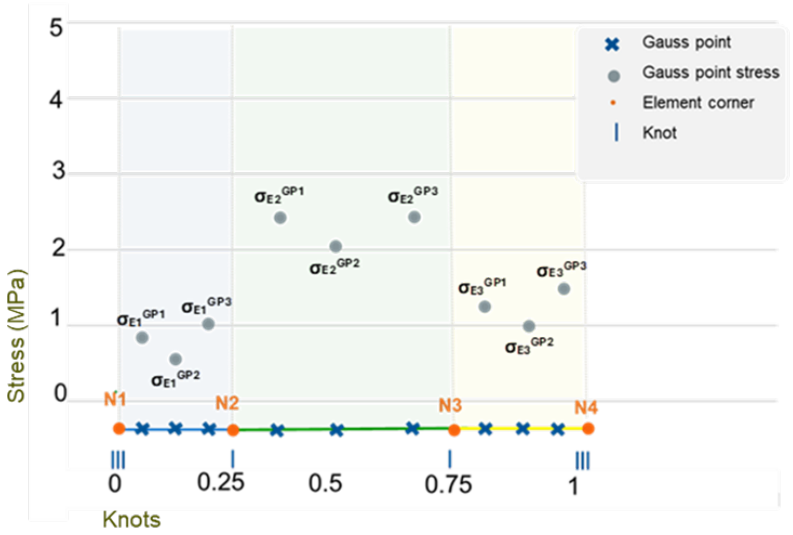


Figure 43: 1D IGA Element: Gauss point Stress vs Knots

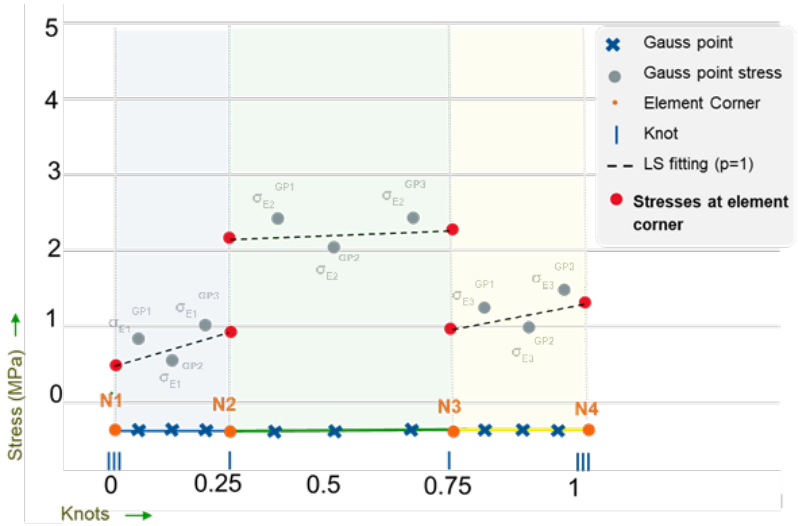


Figure 44: C^{-1} continuous stress across the element after using LSM fitting

In Figure 45 the Gauss-point stresses are fitted using a least-squares method to obtain approximated stress values at the element corners.Finally, in Figure 46 by averaging the corner stresses from adjacent elements, continuity is restored , resulting in a smooth and physically meaningful stress field.

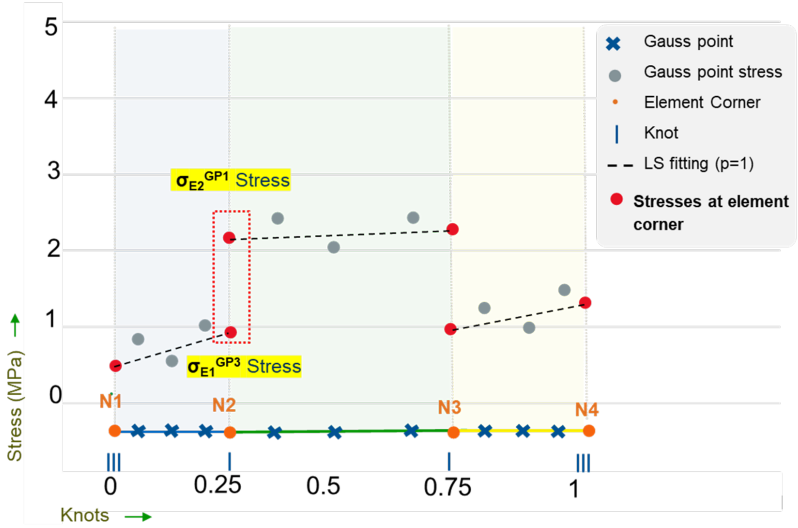


Figure 45: C^{-1} continuous stress due to 2 stress value at the same element corner.

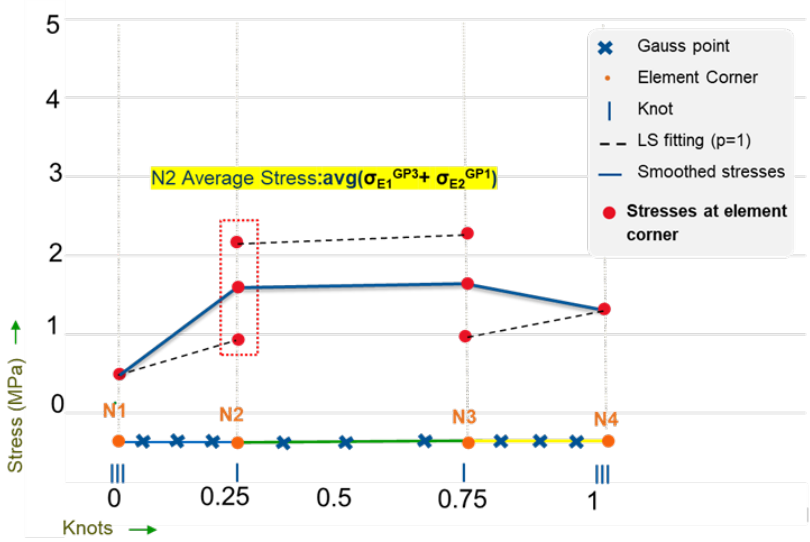


Figure 46: C^0 continuous stress across the element after using averaging method

3.2.4 Code Overview

The following flowchart 47 summarizes how the main function for Extrapolated_Nodal_Stresses(RS) operates when it is called in the main function after the analysis step. The function takes hierarchical space object as input.Further, the following parameters can be used to visualize the results through the existing spline visualization plot_trimmed_surface(RS, n_points=10, s=0.05, evalpnt=None, stressinterp=False, output_path=None, display_mode=None) where display_mode can be set to one of the followings:

- nodal_sigma_xx : visualizes the least-squares fitted σ_{xx} stress, showing inter-element smooth stress.
- smoothed sigma_xx : visualizes the weighted-averaged σ_{xx} across elements, providing a smoother stress distribution.
- nodal stress_von_mises : visualizes the least-squares fitted vM stress, showing inter-element smooth stress.
- smoothed stress_von_mises : visualizes the weighted-averaged vM across elements, providing a smoother stress distribution.

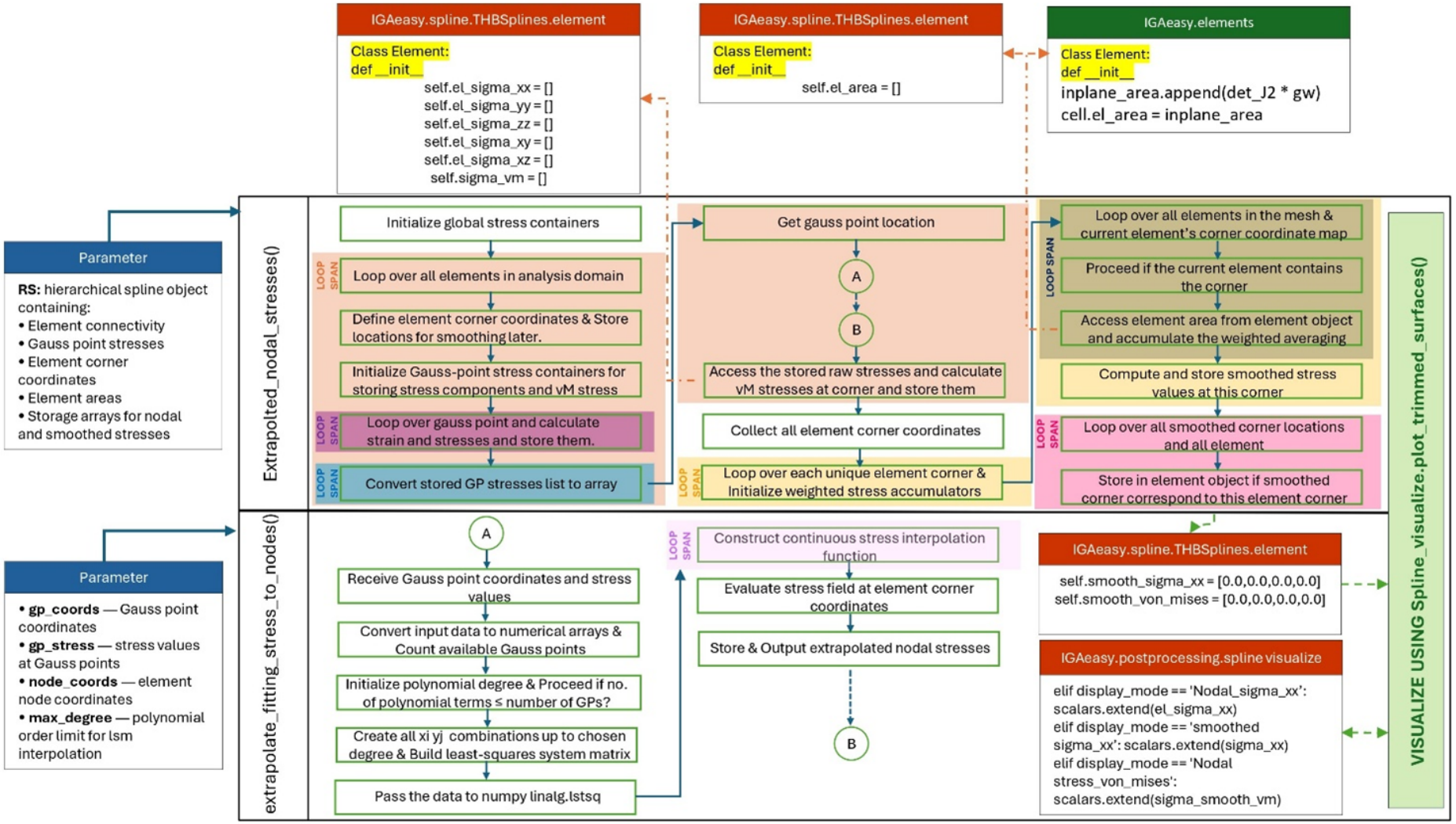


Figure 47: Stress Smoothening function working

3.3 Results

3.3.1 Stress Smoothening

The following section presents the numerical results obtained from the developed stress-smoothening and visualization procedure. For each test problem, the stress distribution is illustrated in three stages: first, the initial visualization based directly on element-level values; second, the inter-element continuous stress field obtained after least-squares fitting at the element corners; and finally, the fully smoothed stress distribution achieved through weighted averaging. This step-by-step presentation highlights the progressive improvement in continuity and visual interpretability of the stress field.

The benchmark problem of the pinched cylinder is analysed using the IGAeasy framework, and the numerical results are validated against reference solutions. Figure 48 shows the convergence diagram for pinched cylinder benchmark.Figure 49 shows the stress smoothening steps.

Continuing further this is results for a example setup of Plate fixed at all edges which was solved and then later visualized using the stress smoothening method. Figure 50 shows the problem setup.Figure 51 shows the stress smoothening steps.

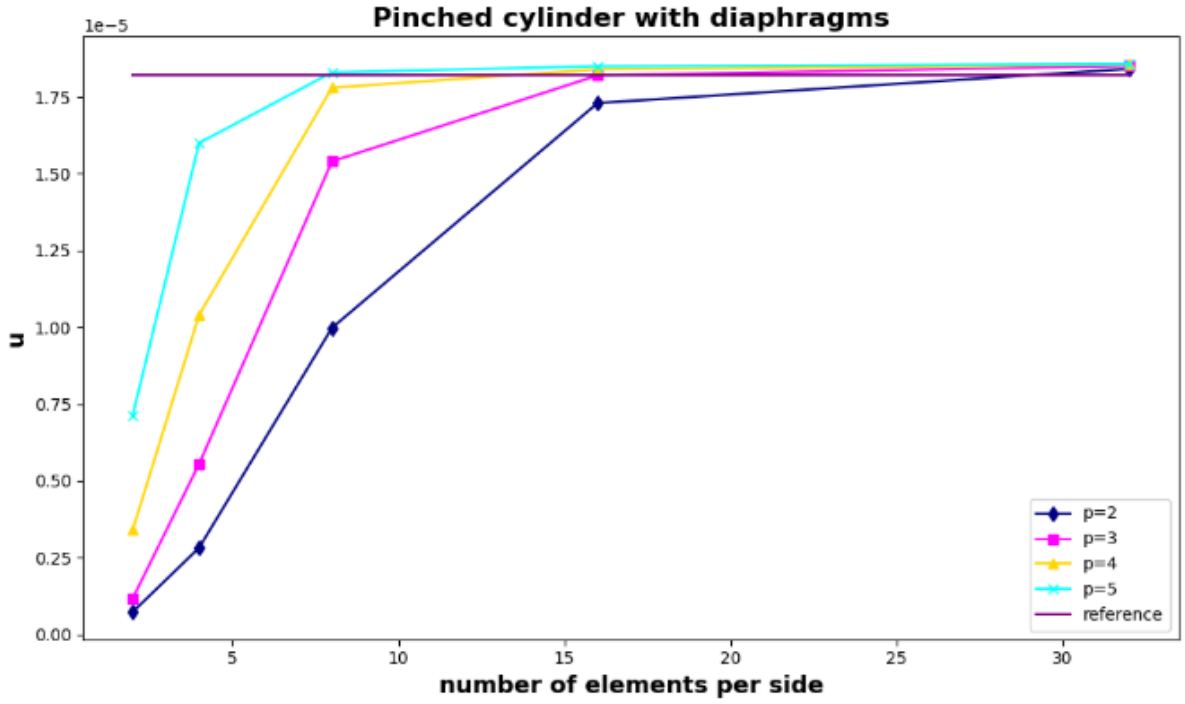
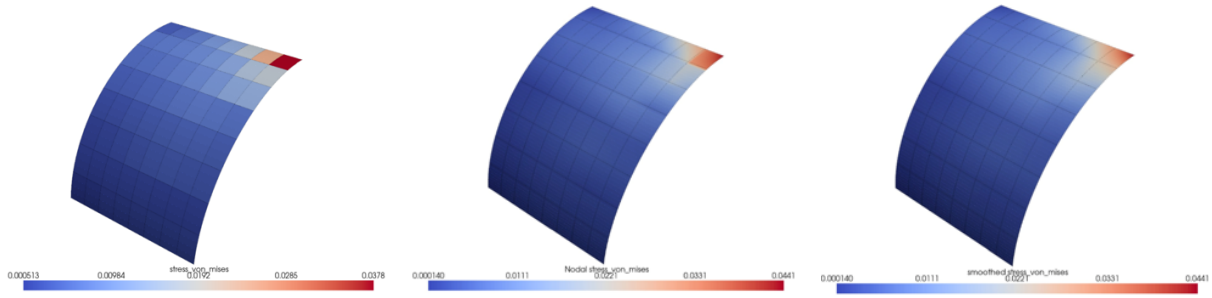
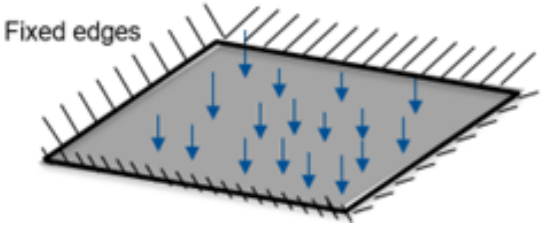


Figure 48: y-displacement of pinched cylinder benchmark



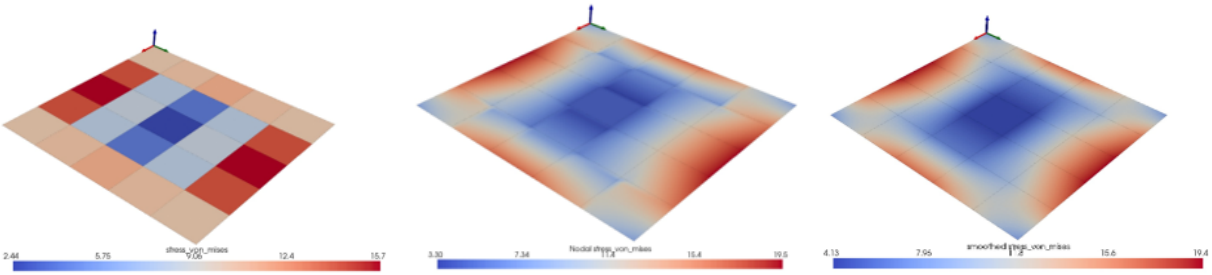
(a) (b) (c)

Figure 49: Stress smoothing comparison: (a) Element wise averaged stress, (b) Inter element smooth stress after LSM fitting, (c) Final visualization after using weighted averaging method.



Length	= 500 mm
Breadth	=500 mm
Thickness	= 1 mm
Load	= 1 N
Youngs modulus	= 3.0E+6 MPa
Poisons ratio	= 0.3
Boundary conditions	=Fixed at all edges

Figure 50: Problem setup: fixed plate under uniform load



(a) (b) (c)

Figure 51: Stress smoothing comparison: (a) Element wise averaged stress, (b) Inter element smooth stress after LSM fitting, (c) Final visualization after using weighted averaging method.

3.4 Conclusion

The aim of stress smoothing was successfully achieved through the proposed process, producing realistic results when compared with the standard spline-based stress quantities and visualizations already available. The method works effectively within the

THB-spline setup and can be applied to geometries that are naturally and easily modeled within the capabilities of IGAeasy package, enabling reliable and smooth stress visualization.

3.5 Future wrork

The proposed stress smoothing method can be further extended to account for trimming and local refinement within the IGA framework. Incorporating these aspects would improve its applicability to more complex geometries and enabling broader and more robust stress visualization capabilities.

References

[1] P. Wriggers, J. Schröder, and A. Schwarz. A finite element method for contact using a third medium. *Computational Mechanics*, 52(4):837–847, October 2013.

[2] Gore Lukas Bluhm, Ole Sigmund, and Konstantinos Poulios. Internal contact modeling for finite strain topology optimization. *Computational Mechanics*, 67(4):1099–1114, April 2021. arXiv:2010.14277 [cs].

[3] Andreas Henrik Frederiksen, Ole Sigmund, and Konstantinos Poulios. Topology optimization of self-contacting structures. *Computational Mechanics*, 73(4):967–981, April 2024.

[4] Josef Kiendl, Kai-Uwe Bletzinger, Johannes Linhard, and Roland Wüchner. Isogeometric shell analysis with kirchhoff–love elements. *Computer Methods in Applied Mechanics and Engineering*, 198(49-52):3902–3914, 2009.

[5] Carlos A. Felippa. Stress recovery. In *Introduction to Finite Element Methods*, chapter 28. University of Colorado, Boulder, CO, 2004. Course Lecture Notes.