Implementation of Trimmed Isogeometric Analysis Membrane Structures

Supervisor:

Andreas Apostolatos

Team:

Bassel Saridar

Cleo Reihl

Dagmawi Bekel

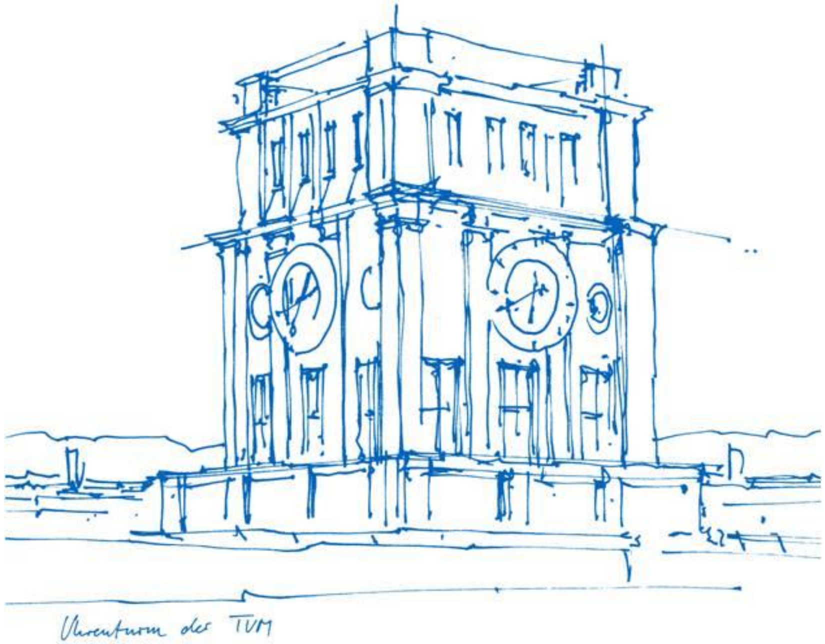
Khaled Boulbrachene

Mahmoud Ammar

Mahmoud Zidan

Muhammad Salman Yousaf

Srikkanth Varadharajan

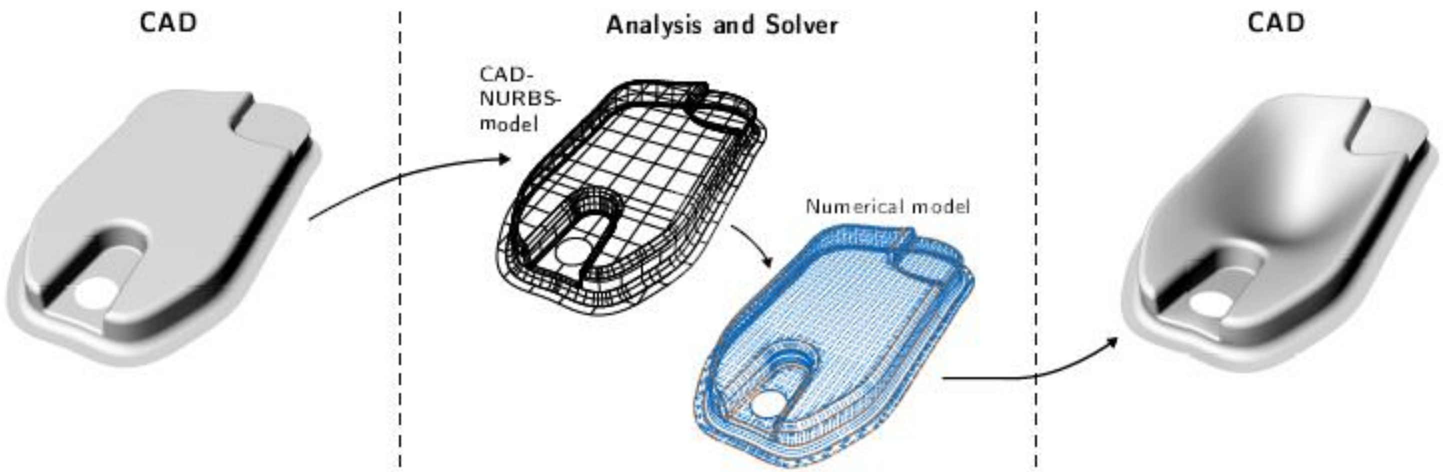


Outline

- Introduction
- Parsing
- Preprocessing
- Triangulation
- Cables
- Fixing Control Points Without Contribution
- Gauss Points for Visualization
- Results

Iso-Geometric B-rep Analysis (IBRA)

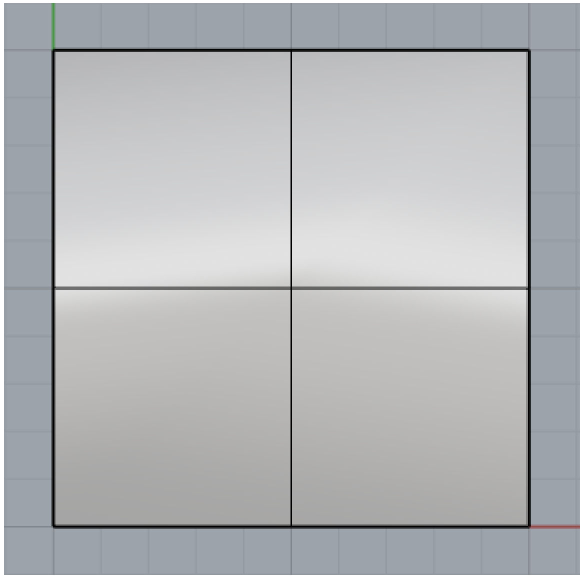
- New Structural Analysis Technique
- Uses Iso-Geometric Analysis Techniques
- Allows direct analysis of CAD models

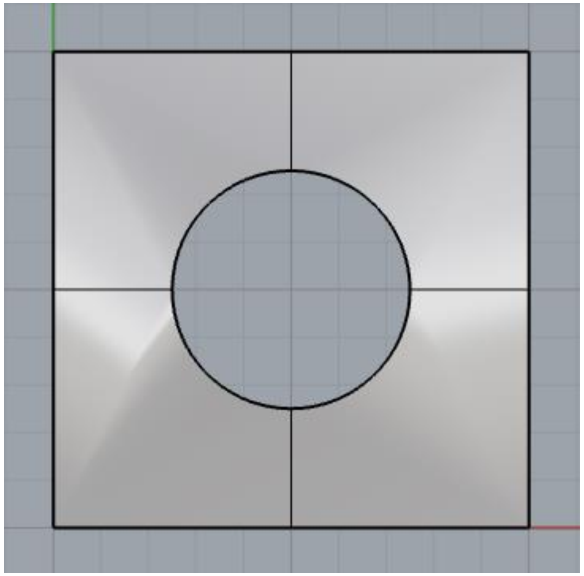


Realization of CAD-integrated Shell Simulation
based on Isogeometric B-Rep Analysis
T. Teschemacher1*, A. M. Bauer1, T. Oberbichler1, M. Breitenberger1, R. Rossi2, R. Wüchner1 and K.-U. Bletzinger1

Trimming Surfaces

- Surfaces Cut-off from original surface
- Curves separating visible to non visible surfaces
- Include their own Parametric Space





Boundary Representation (B-rep)

- Used in CAD
- Describes object by its Skin
- Components:
  - Faces = Surfaces
  - Edges = Curves
  - Vertices = Points

V	Vertices
E	Edges (direction is not considered)
C	Space curves
Č	Trimming curves
S	Surfaces
$\mathcal{D}$	Trimmed domain

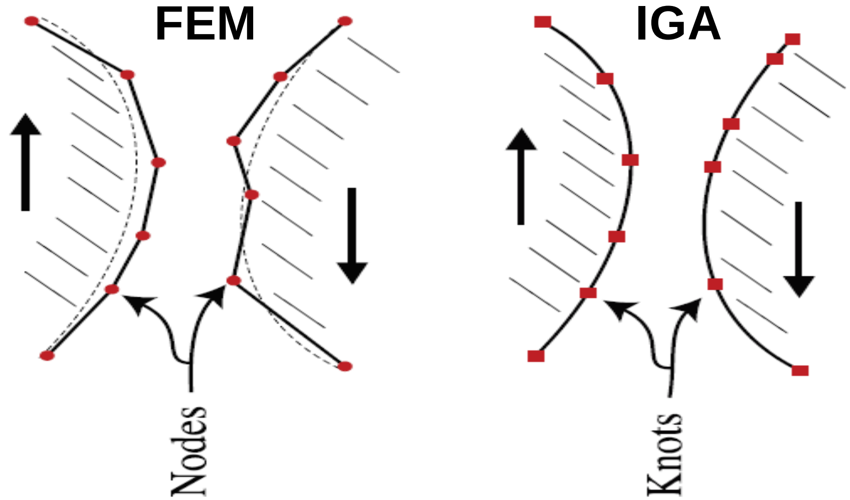
Motivation for IGA

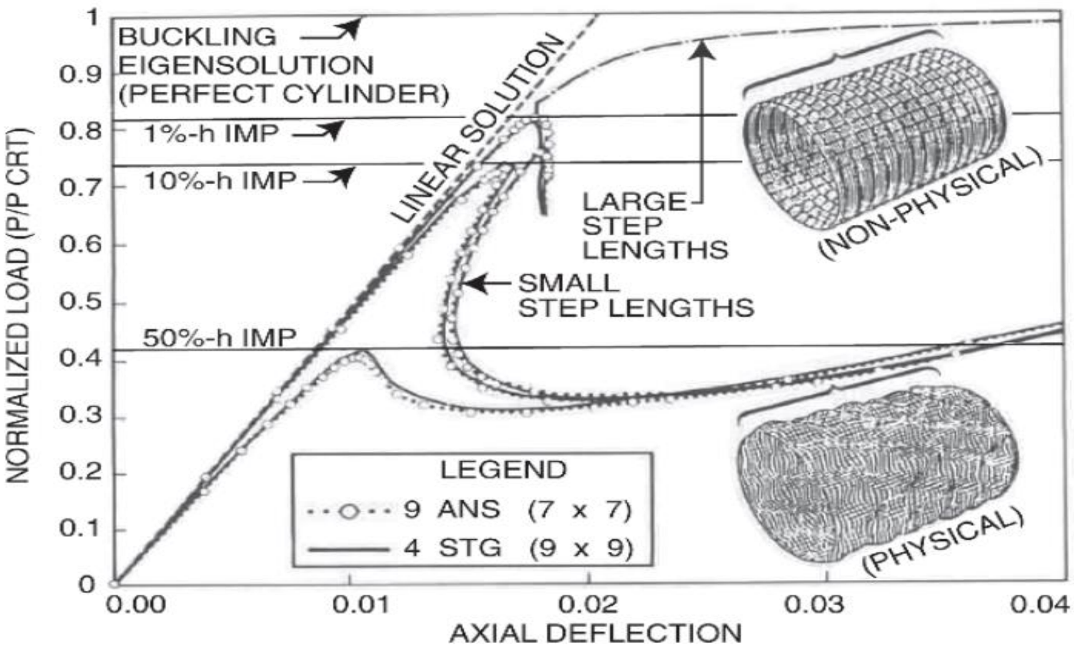
Smooth geometry requirement

1. Contact

2. Shell buckling

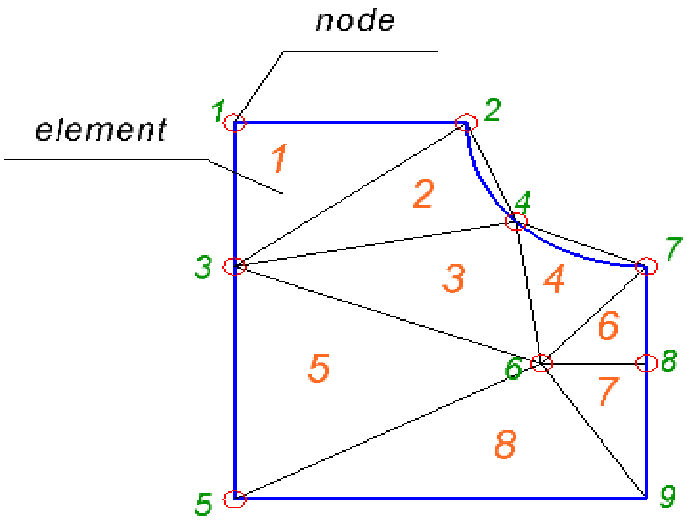
3. Boundary layer phenomenon



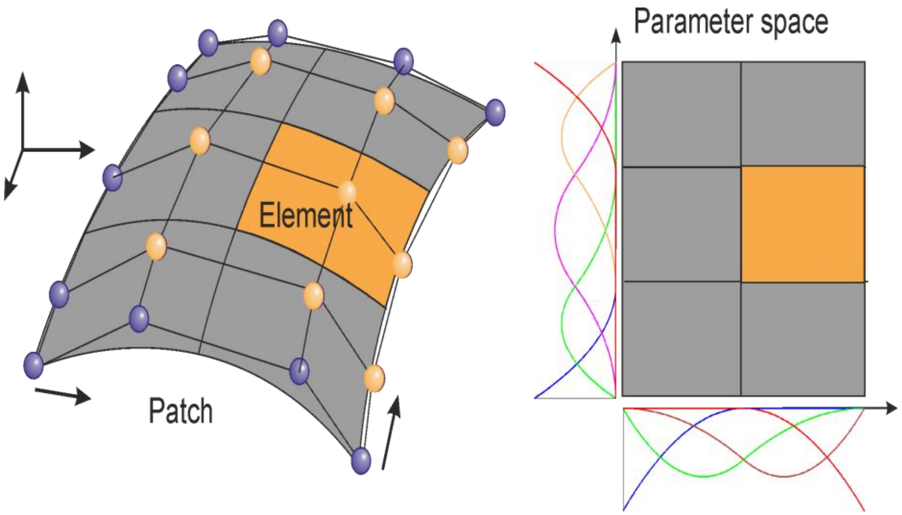


Comparison between FEM and IGA

1. Element definition



FEM

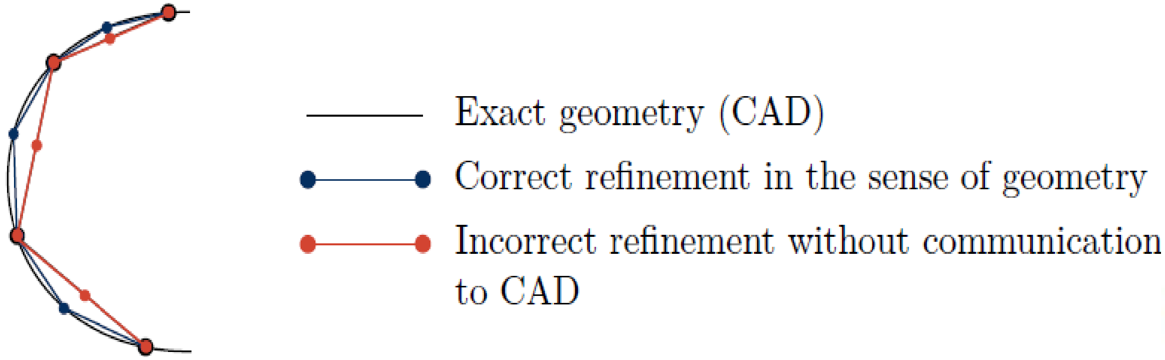


IGA

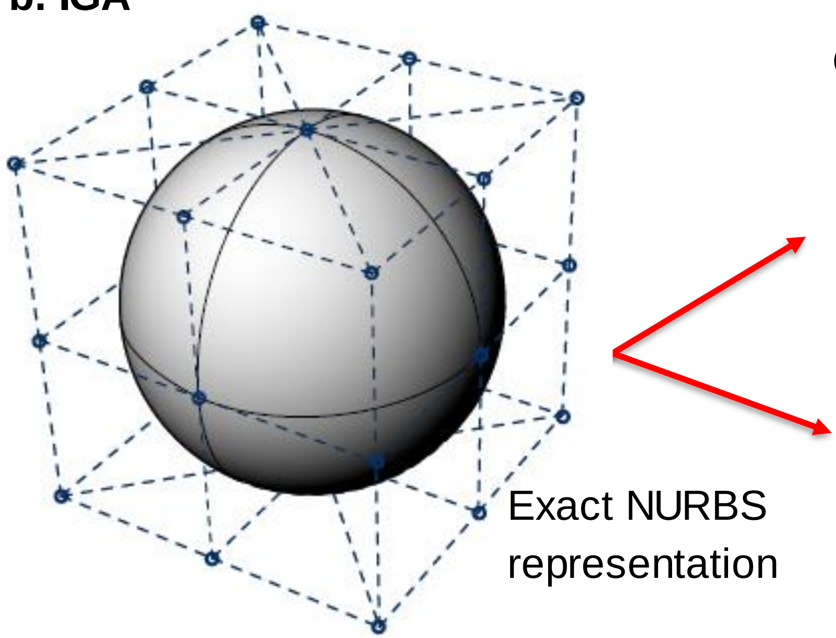
Comparison between FEM and IGA

2. Refinement

a. FEM

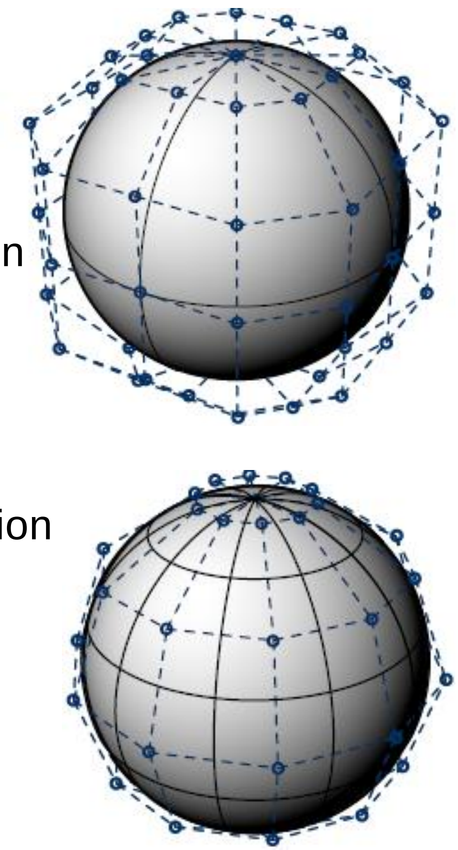


b. IGA



Order elevation

Knot insertion



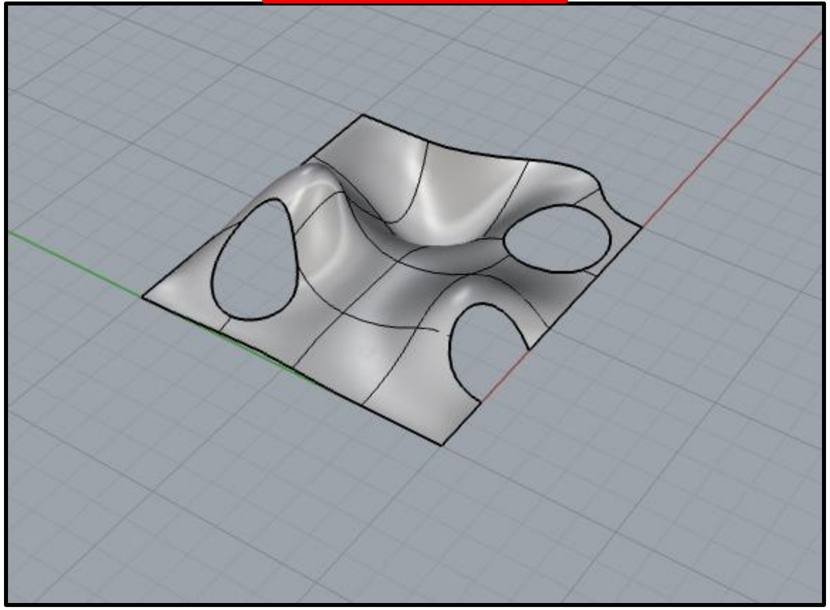
Outline

- Introduction
- Parsing
- Preprocessing
- Triangulation
- Cables
- Fixing Control Points Without Contribution
- Gauss Points for Visualization
- Results

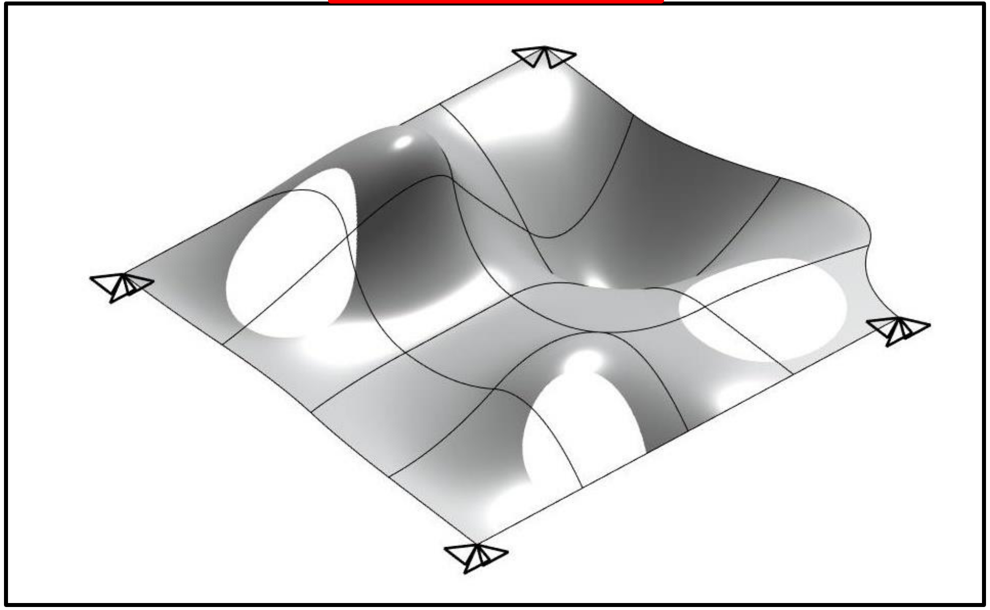
Parsing

- Create the CAD geometry in Rhino.
- Using TeDa Plugin, Create a JSON file.
- Read JSON file in Matlab.

Rhino



Matlab



JSON File Structure

Example:

Polynomial Degree

↓

p_surface = val.breps.faces.surface.degrees(1)

Knot Vector

↓

Xi_surface = val.breps.faces.surface.knot_vectors{1,:}';


```json
{
    "breps": [{
        "brep_id": 1,
        "faces": [{
            "brep_id": 2,
            "swapped_surface_normal": false,
            "surface": {
                "is_trimmed": true,
                "is_rational": false,
                "degrees": [1, 1],
                "knot_vectors": [
                    [0, 10],
                    [0, 10]
                ],
                "control_points": [
                    [1, [0, 0, 0, 1]],
                    [2, [10, 0, 0, 1]],
                    [3, [0, 10, 0, 1]],
                    [4, [10, 10, 0, 1]]
                ]
            },
            "boundary_loops": [{
                "loop_type": "outer",
                "trimming_curves": [{
                    "trim_index": 0,
                    "curve_direction": true,
                    "parameter_curve": {
                        "is_rational": false,
                        "degree": 1,
                        "knot_vector": [0, 10],
```

JSON Parser

JSON Parser function was implemented in order to return the parsed geometry information from the JSON File created with Rhino 5

Input
Folder Name Containing the JSON File

Output	
1	Polynomial Degree of the untrimmed Patch in Xi-direction
2	Polynomial Degree of the untrimmed Patch in Eta-direction
3	Knot vector of the untrimmed Patch in Xi-direction
4	Knot vector of the untrimmed Patch in Eta-direction
5	Control Points of the untrimmed Patch
6	Trimming curves structure

Trimming curves structure

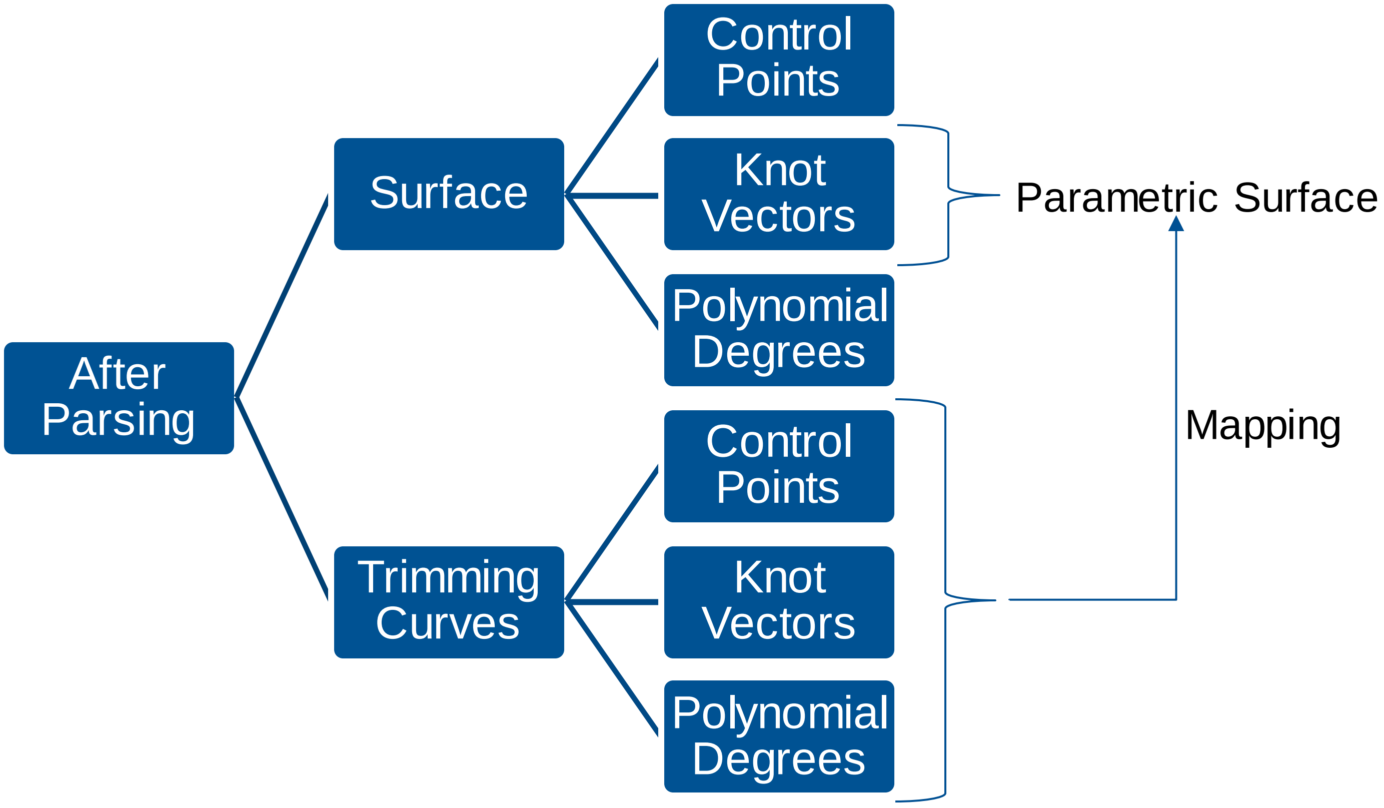
Structure containing all trimming curve information

Trimming curves structure	
1	Number of curves in the outer boundary loop
2	Polynomial Degree of the outer trimming curve
3	Knot vector of the outer trimming curve
4	Control points of the outer trimming curve defined in counter-clockwise direction
5	Number of inner boundary loops
6	Knot vector of the inner trimming curve
7	Control points of the inner trimming curve defined in clockwise direction)

Outline

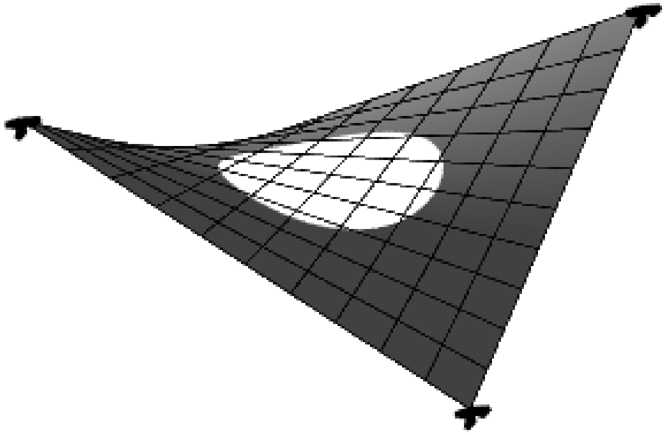
- Introduction
- Parsing
- Preprocessing
- Triangulation
- Cables
- Fixing Control Points Without Contribution
- Gauss Points for Visualization
- Results

Preprocessing



Preprocessing (cont'd)

Parametric Surface → Evaluation Points → Eliminating Outside Evaluation Points → Mapping to Physical Surface

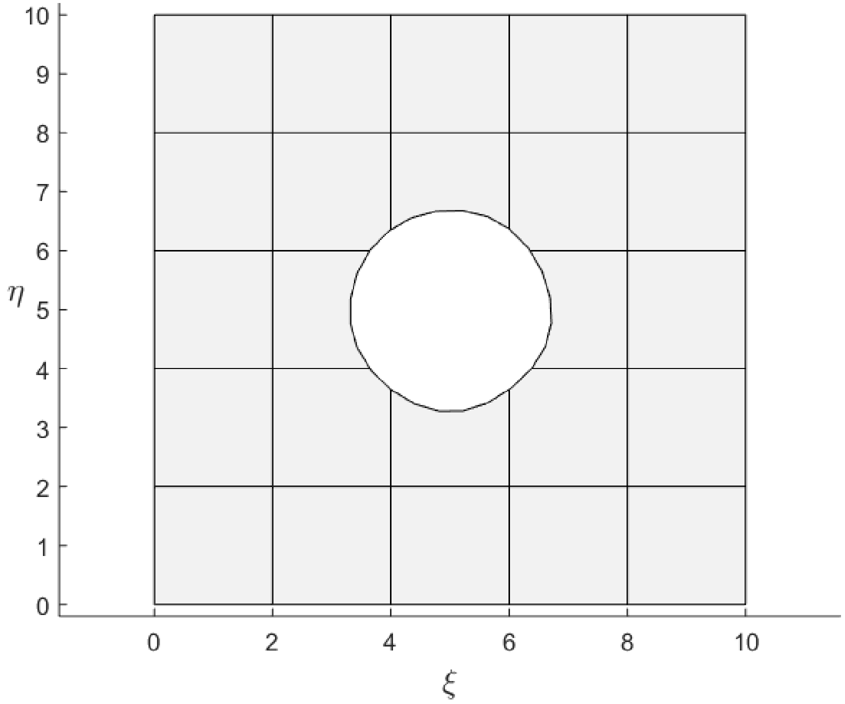


Outline

- Introduction
- Parsing
- Preprocessing
- Triangulation
- Cables
- Fixing Control Points Without Contribution
- Gauss Points for Visualization
- Results

Triangulation

In order to do the analysis of the trimmed isogeometric patch Gauss points have to be distributed



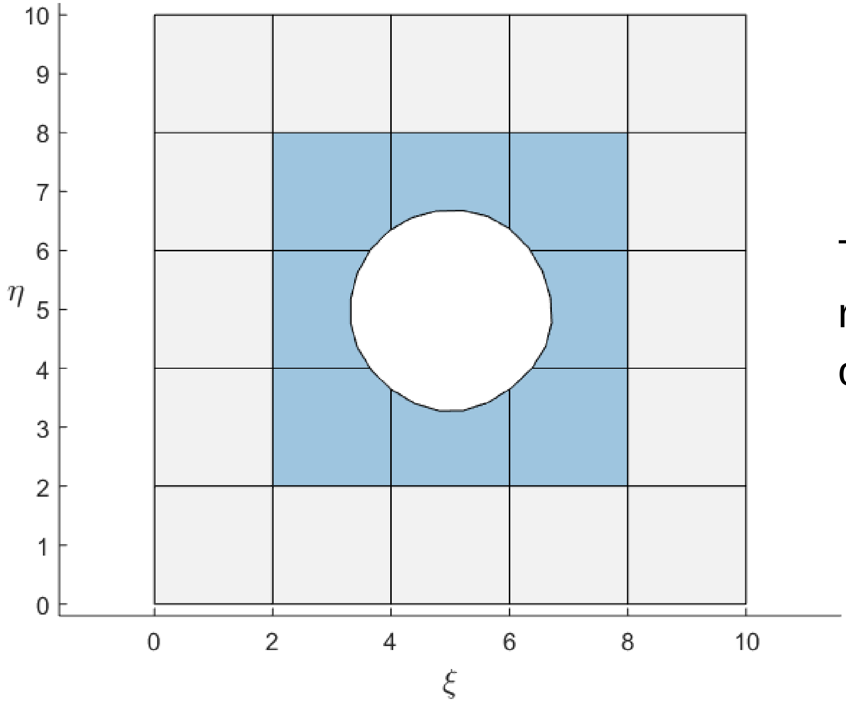
Parameter domain

Steps:

1. Find the intersected Elements
2. Triangulate intersected elements
3. Distribute Gauss Points in not intersected elements
4. Distribute Gauss Points in the intersected elements

Triangulation (cont‘d)

1. Find the elements of the patch that are being intersected by the boundary loop

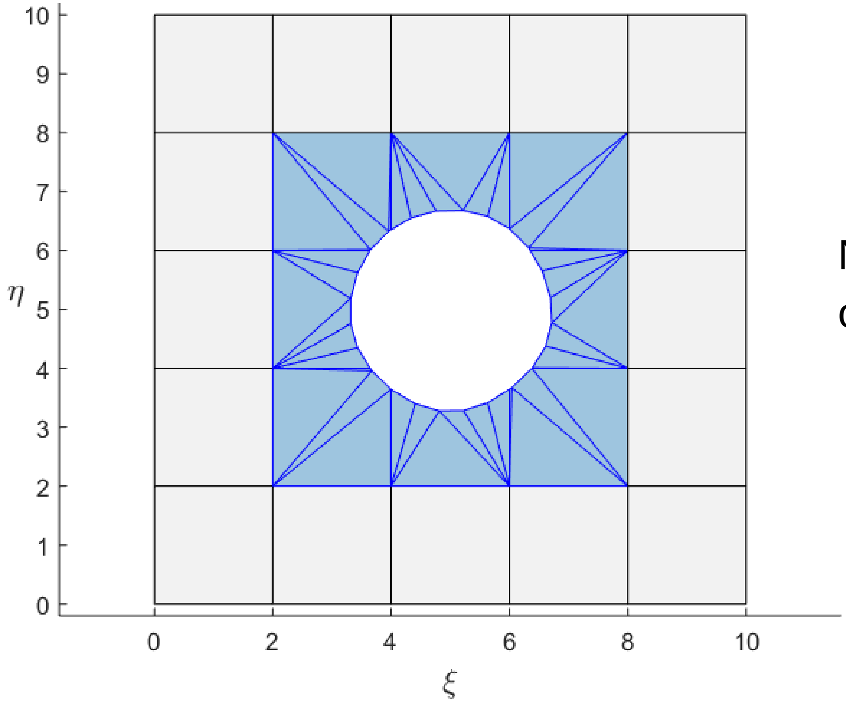


Parameter domain

The boundary loop is not a smooth curve, but rather a polygon of the discrete points lying on the trimming curve

Triangulation (cont‘d)

2. Triangulate the elements that are being intersected by the inner trimming curve using the constrained Delaunay triangulations



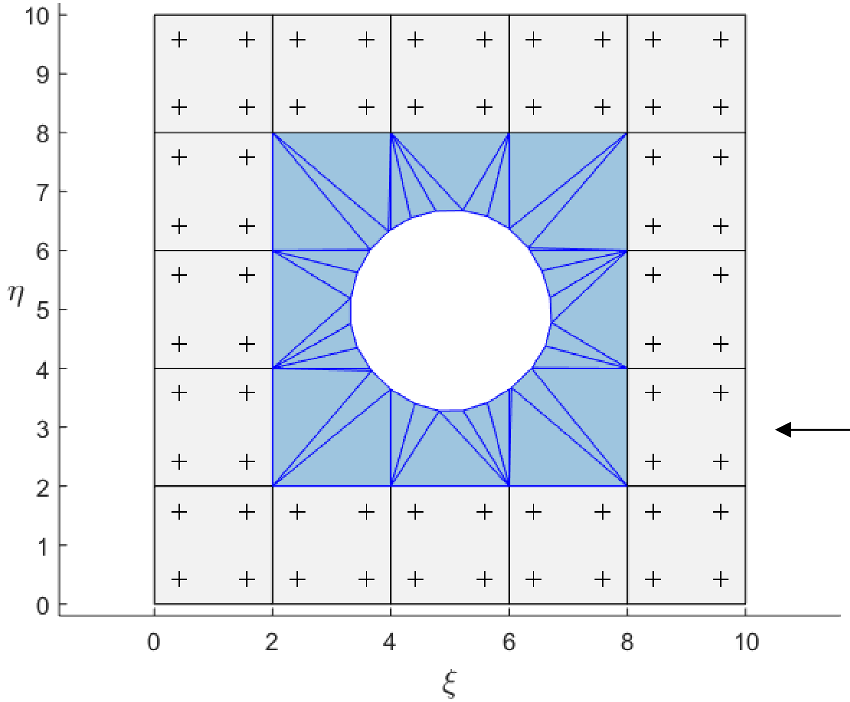
Parameter domain

Number of triangles depends on the number of discrete point on the trimming curve

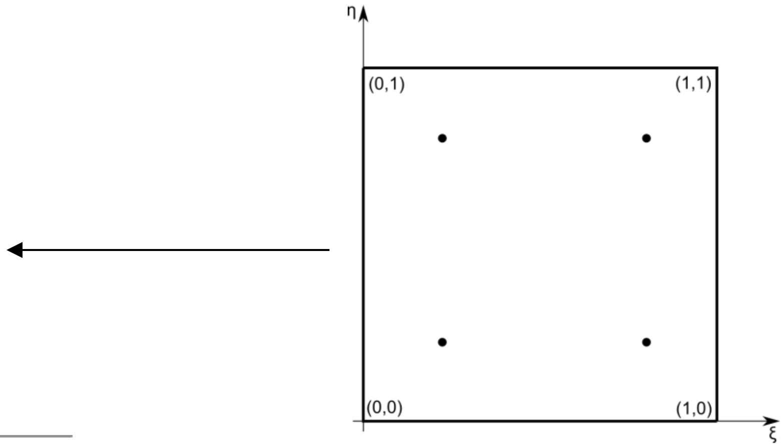
Triangulation (cont‘d)

3. Elements that are not intersected by the trimming curve will be filled according to a quadrilateral rule

Linear Mapping between Gauss domain and Parameter domain



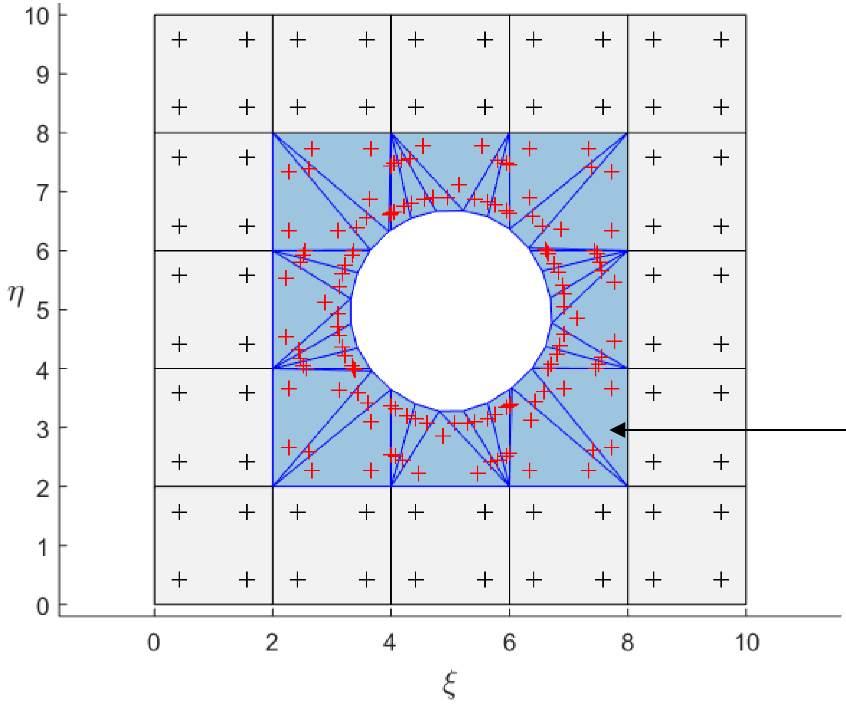
Parameter domain



Gauss domain

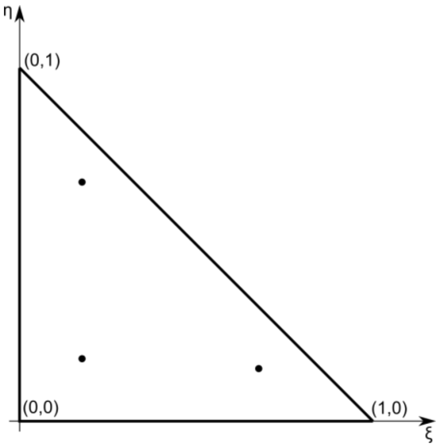
Triangulation (cont‘d)

4. Intersected elements have to be filled with Gauss Points according to a canonical triangle rule



Parameter domain

Linear Mapping between Gauss domain and Parameter domain

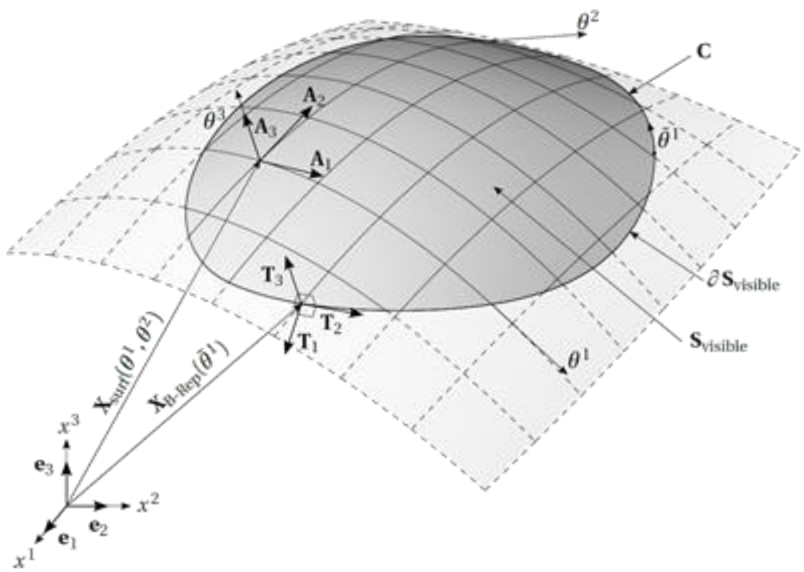


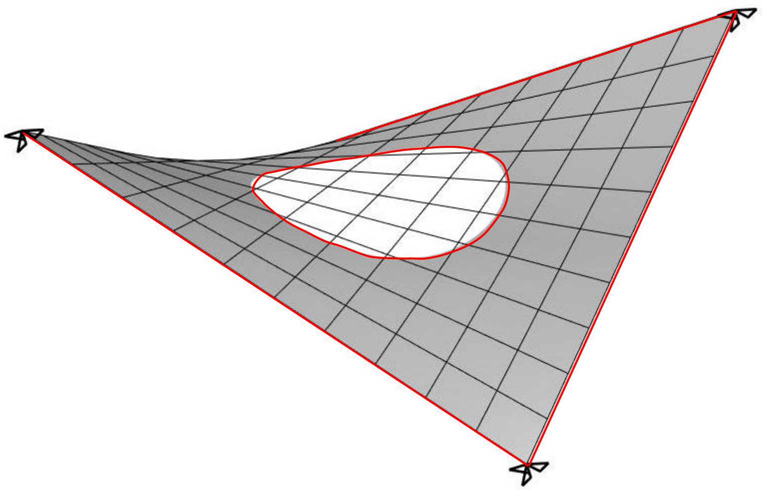
Gauss domain

Outline

- Introduction
- Parsing
- Preprocessing
- Triangulation
- Cables
- Fixing Control Points Without Contribution
- Gauss Points for Visualization
- Results

Cables



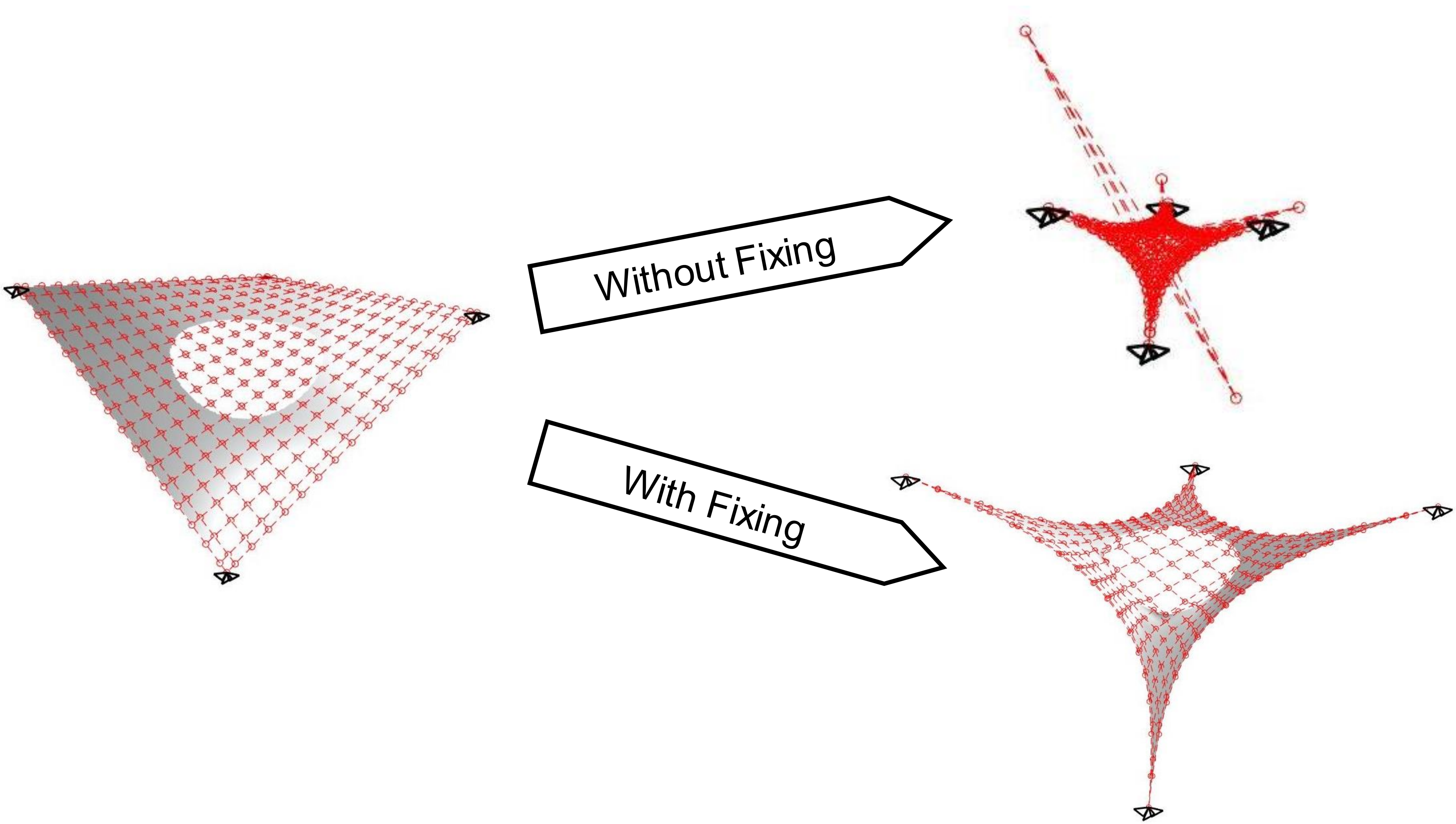


For membrane theory to be applied, we are adding extra stiffness on the boundaries (cables)

$$\mathbf{A_1} = \frac{\partial \mathbf{X_{surf}}}{\partial \theta^1} ; \mathbf{A_2} = \frac{\partial \mathbf{X_{surf}}}{\partial \theta^2}$$

$$\mathbf{T_{trimming}} = \frac{\partial \mathbf{X_{surf}}}{\partial \tilde{\theta}} = \frac{\partial \mathbf{X_{surf}}}{\partial \theta^1}\frac{\partial \theta^1}{\partial \tilde{\theta}} + \frac{\partial \mathbf{X_{surf}}}{\partial \theta^2}\frac{\partial \theta^2}{\partial \tilde{\theta}} = \mathbf{A_1}\breve{T}_1 + \mathbf{A_2}\breve{T}_2$$

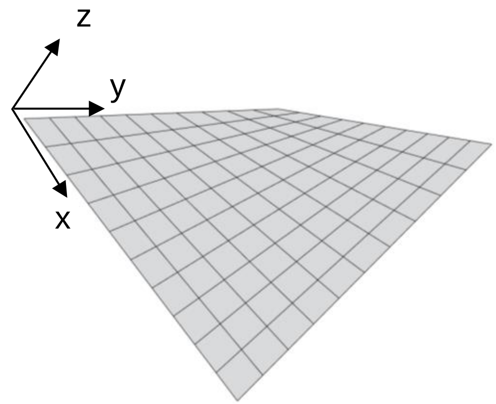
Fixing Control Point without Contribution



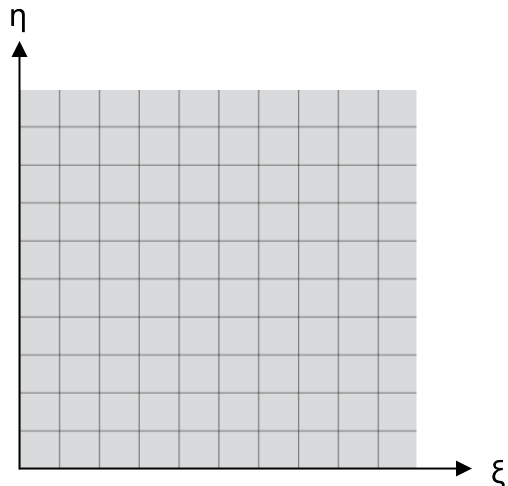
Outline

- Introduction
- Parsing
- Preprocessing
- Triangulation
- Cables
- Fixing Control Points Without Contribution
- Gauss Points for Visualization
- Results

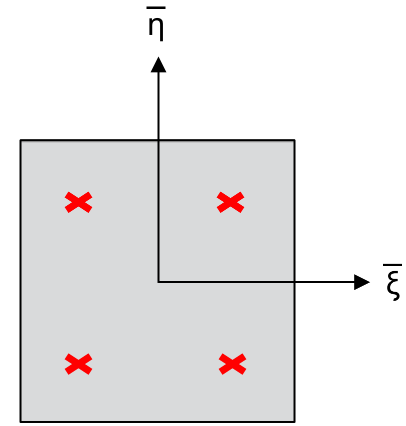
Gauss Points for visualization?



Physical space (x,y,z)



Parametric space of surface (ξ,η)

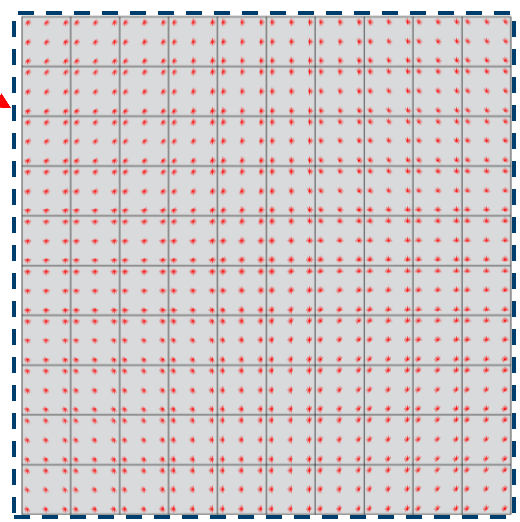


Gaussian space ($\bar{\xi}$,$\bar{\eta}$) E [-1,1]

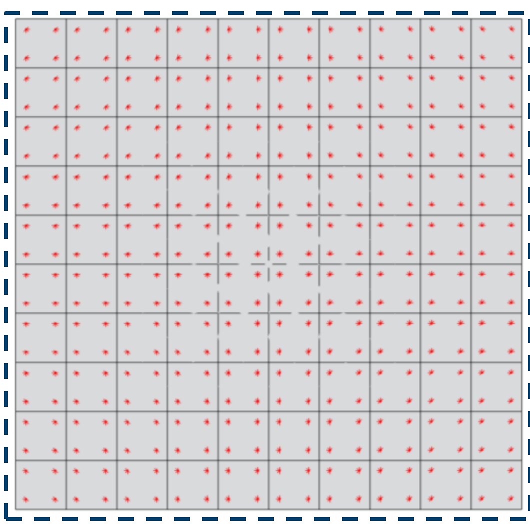
✖ Gauss points

Problem : No points on the edges of the structure !!

Solutions :
1. Add evaluation point to visualize the edges
2. Use the GPs from the cables running along the edges

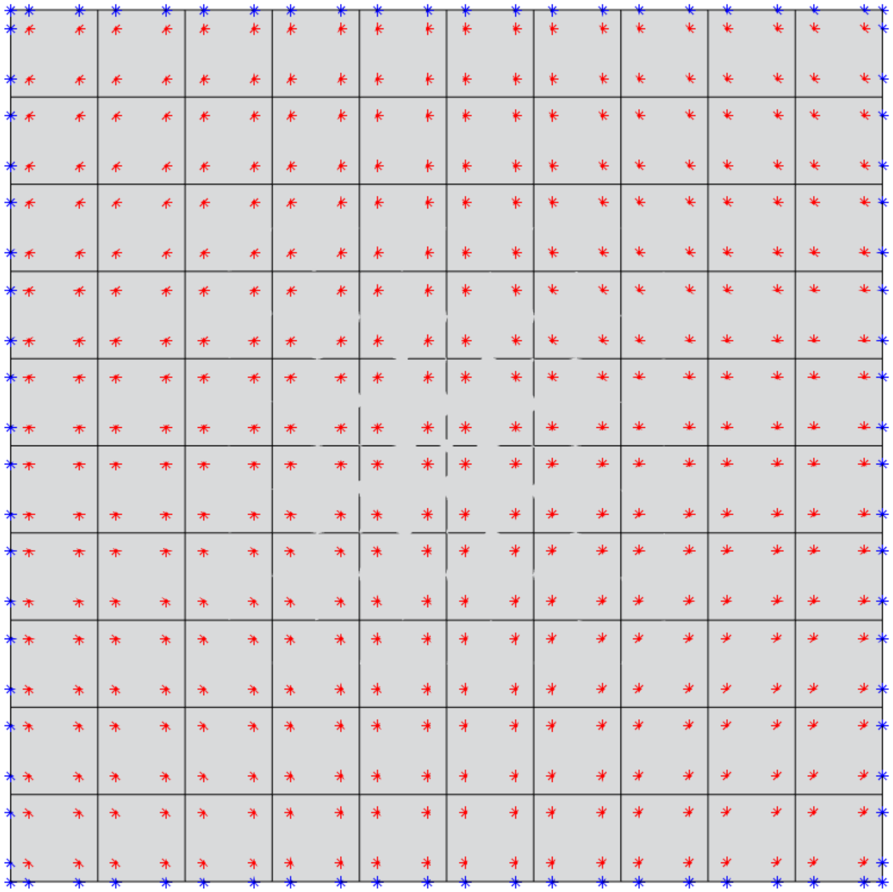


3 GPs per direction



2 GPs per direction

Points at the edges of the structure

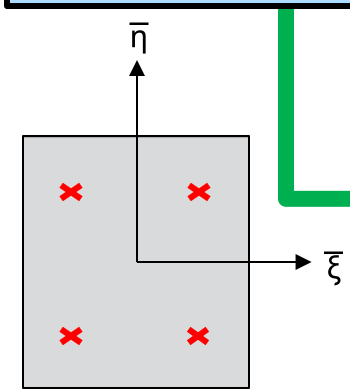


* Evaluation points at the boundaries

* Gauss points corresponding to the element

Gauss Points for visualization – How to go about it?

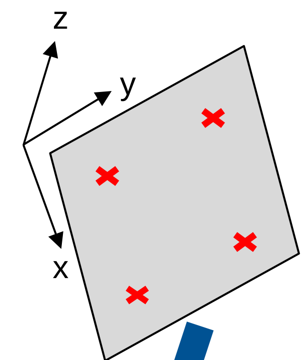
We have $P(\bar{\xi}_i, \bar{\eta}_i)$

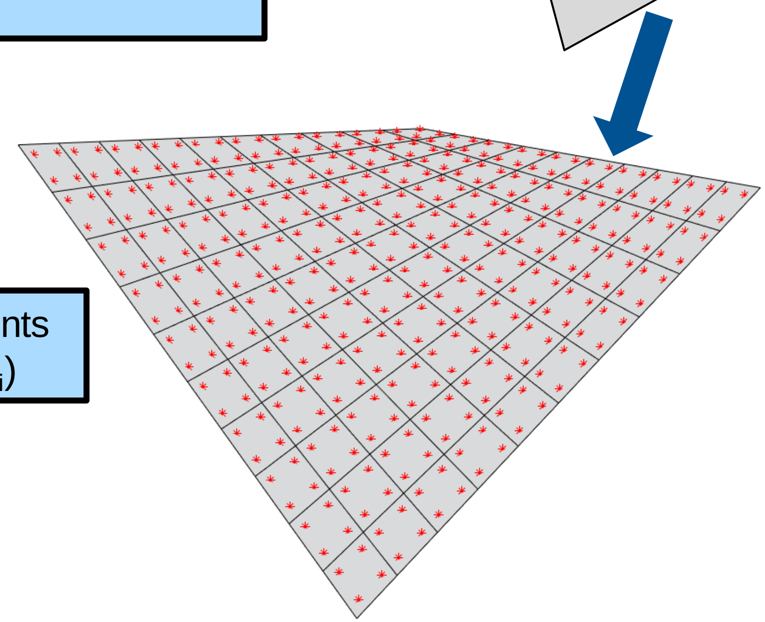


Compute corresponding co-ordinates in parametric space of surface (ξ,η)

Compute basis functions $N_k(\xi)$ and $N_l(\eta)$ at (ξ_i, η_i). Multiply basis functions with CPs of the given element

We get $P(x,y,z)$





Knot spans (ξ_{i+1}, ξ_i) and (η_{i+1}, η_i) of the element for which the GPs are known

(ξ_i, η_i)

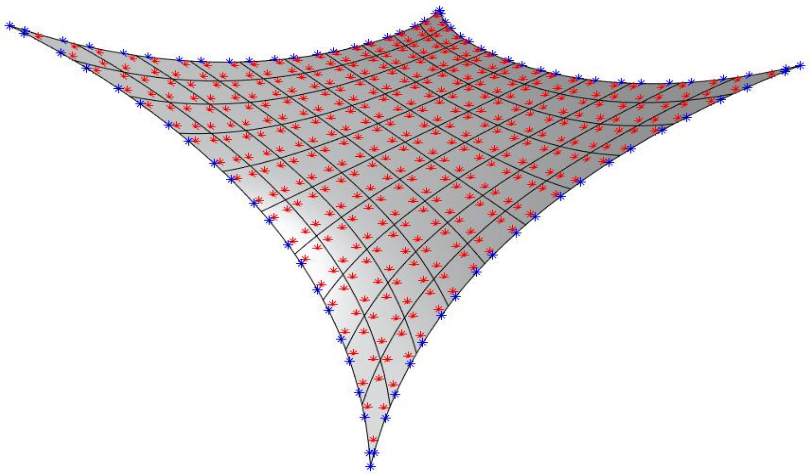
Control points $B(x_i, y_i, z_i)$

$$\xi = \xi_i + (\tilde{\xi} + 1)\frac{(\xi_{i+1} - \xi_i)}{2} = \frac{((\xi_{i+1} - \xi_i)\tilde{\xi} + (\xi_{i+1} + \xi_i))}{2},$$

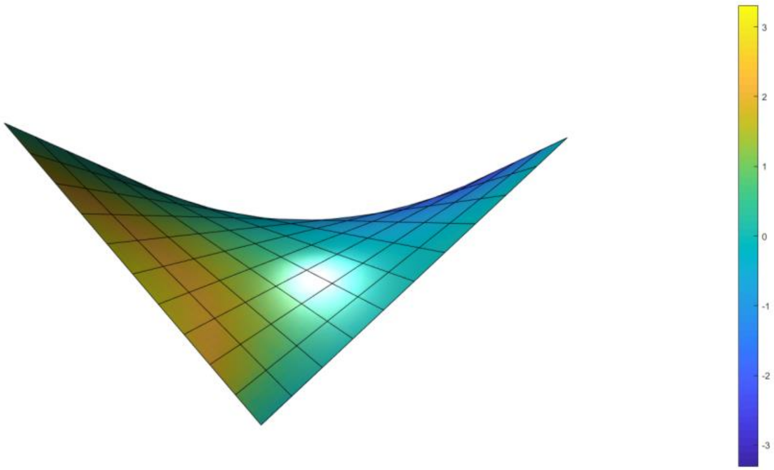
$$\eta = \eta_j + (\tilde{\eta} + 1)\frac{(\eta_{j+1} - \eta_j)}{2} = \frac{((\eta_{j+1} - \eta_j)\tilde{\eta} + (\eta_{j+1} + \eta_j))}{2},$$

Source : Cottrell, Hughes, Bazilevs

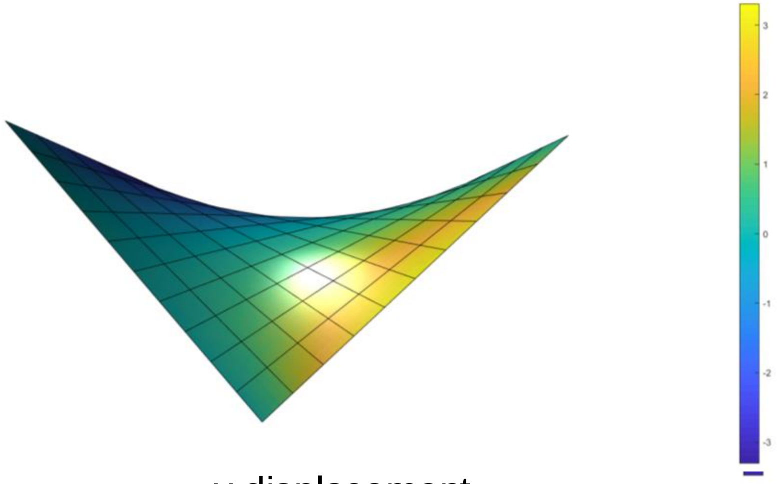
Membrane after form finding



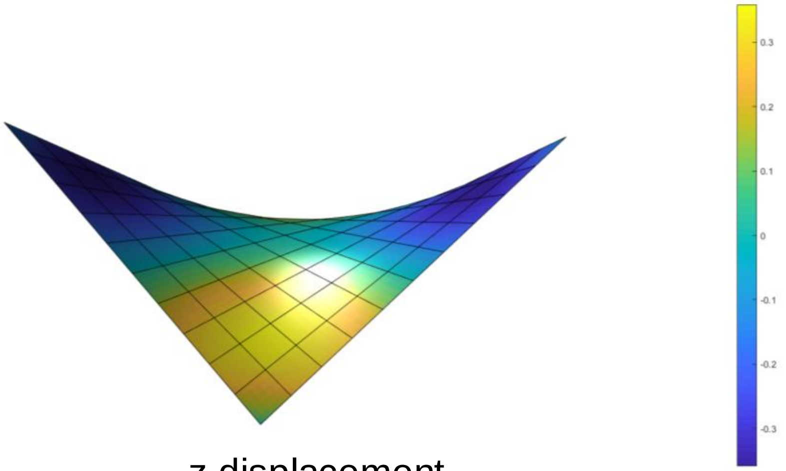
Membrane after form finding



x-displacement
Plotted with respect to initial configuration



y-displacement
Plotted with respect to initial configuration



z-displacement
Plotted with respect to initial configuration

Outline

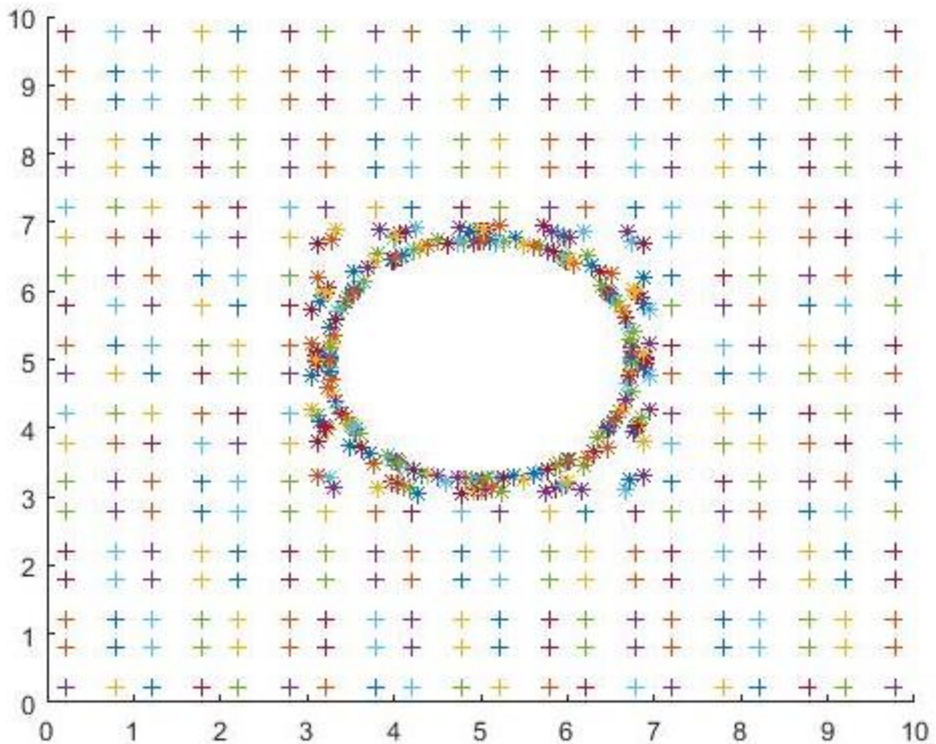
- Introduction
- Parsing
- Preprocessing
- Triangulation
- Cables
- Fixing Control Points Without Contribution
- Gauss Points for Visualization
- Results

Postprocessing

After trimming the Gauss points are no longer distributed grid wise

→ Surf command cannot be used

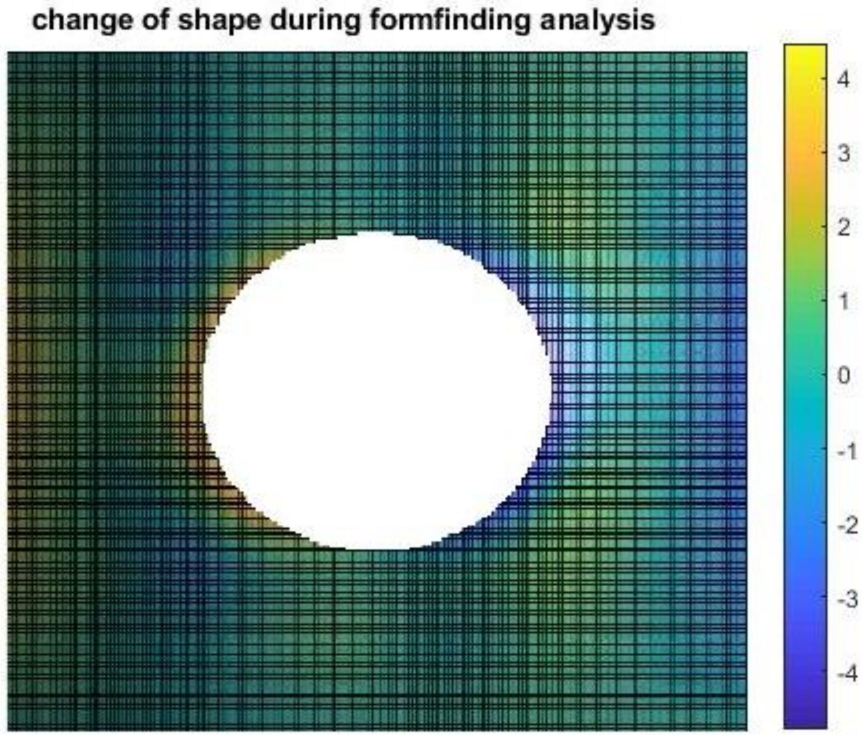
→ Use plotting command for scattered data or use evaluation points instead of Gauss points



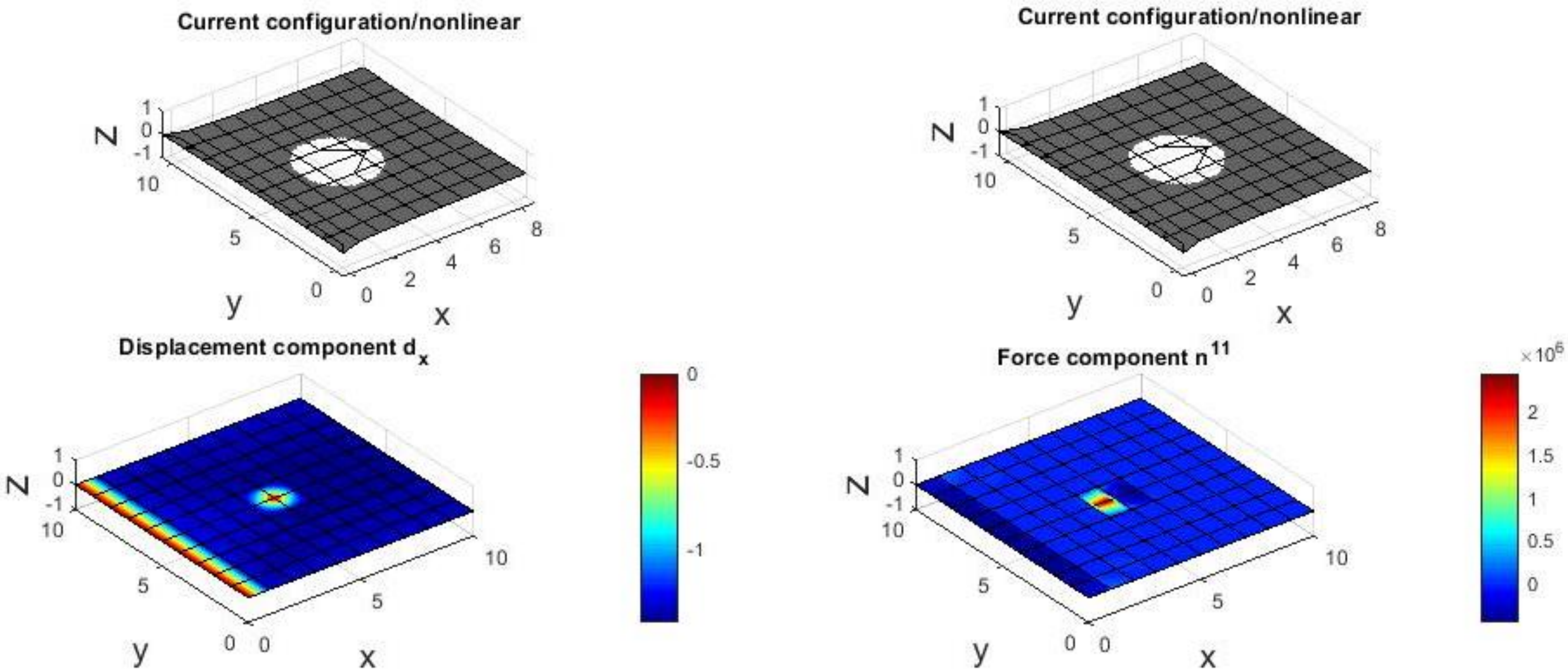
Postprocessing – form-finding

show changes of the shape (e.g. shifting of x-coordinate) during the form-finding process:

sum up displacements of control points in every iteration step,
compute corresponding displacements of membrane surface



Postprocessing – nonlinear analysis



Next steps

- Make the code work correctly / Find the bug
- Validate the results using a commercial code (e.g. Ansys)
- Write the report

Thank you for your attention

Any Questions?