Isogeometric analyis of trimmed membrane structures

Mahmoud Ammar[1], Dagmawi Bekel[1], Khaled Boulbrachene[1], Cleo Reihl[2], Bassel Saridar[1], Srikkanth Varadharajan[1], Salman Yousaf[1], Mahmoud Zidan[1]

[1] M.Sc. student in Computational Mechanics
[2] M.Sc. student in Civil Engineering
Technical University of Munich

ABSTRACT

This paper aims to report the background information, the work flow and the results of a group project in the summer term 2018 at the Chair of Structural Analysis at Technical University of Munich supervised by Andreas Apostolatos. The task was to extend a given code for the IGA of untrimmed membranes to the application of trimmed IGA. IGA uses the CAD information as an analysis model, instead of constructing a separate computational model. In order to get accurate results, a k-refinement is performed on the geometrical model. To represent complex geometries trimmed NURBS geometries are employed. In this present work, form finding is done on trimmed membrane surfaces. This project is implemented using MATLAB *for the analysis and Rhino to define the geometries.*

Keywords: Isogeometric Analysis, Computer Aided Design, Trimmed Geometry, Form Finding

1 Introduction

One of the major challenges for realizing an accurate and reliable design for any type of engineering problem is the interface between the design and the analysis of a structure. It is caused by having different geometry definitions in a variety of softwares. In Computer Aided Design (CAD) smooth geometry descriptions are used, whereas analysis methodologies like the Finite Element Method (FEM) usually use linear basis functions. Before the analysis can be done, the geometriy decription used for the design has to be transferred to the geometry description used for the analysis via meshing. This causes a discretization error.

To overcome this, NURBS-based Isogeometric Analysis (IGA) is employed, which uses the same model for the geometry and the analysis. The advantage of using NURBS basis functions is that they have high continuity, can be refined without changing the original geometry shape and have shown good properties for analysis. After having introduced IGA in some analysis, it turns out that this method is robust and has excellent approximation power.

The process of using NURBS for the geometry description, analysis and post-processing is now named Analysis in Computer Aided Design (AiCAD). In addition to using IGA, also the boundary representation (B-rep) description of the model can be used, which is called Isogeometric Boundary Representation Analysis (IBRA). It avoids the transition from Geometry to Analysis level during preprocessing and back to final geometry during the postprocessing phase [2]. In this report, this concept will be discussed in detail for Membrane Elements, in addition to its use for trimmed membranes since in most engineering design, these structures are more frequently used. A brief description of the AiCAD and the Finite Element Method is shown in Figure 1.

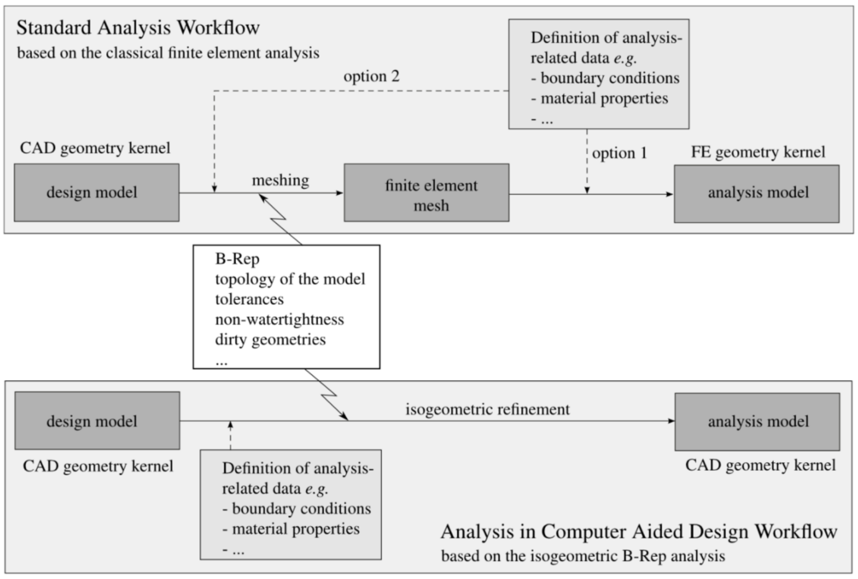


Figure 1: Difference between FEM and AiCAD [2]

In membranes, the external and internal loads are transferred to the supports through tension with neither compression nor bending. This allows membranes to be very thin. Moreover, prestress in both membranes and cables is essential to guarantee stability [10]. The stresses in membrane surfaces are shown in Figure 2.

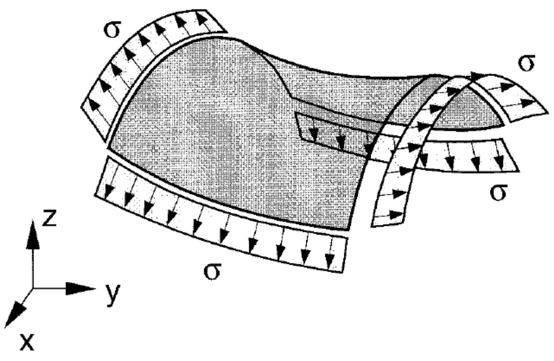


Figure 2: Tangential Surface Stress Field [1]

To design the initial shape of tensile structures it is important to consider the mechanical behavior such that in the unknwon reference configuration the equilibrium is satisfied. To find this geometry, form finding is used regarding the prestress and the boundary conditions. In more detail, form finding is an inverse problem, which determines the equilibrium state of the structure referring to the given stress distribution and boundary conditions, while optimization focuses on finding the optimal geometry capable of bearing the given loads and displacements [1, 10].

Aspects relevant to geometrical definition of the membrane, including the connection between Rhino and MATLAB, form-finding, trimming, triangulation, cable insertion for membrane analysis and post-processing are discussed in this report.

2 Theory

2.1 Geometric Fundamentals

Curves and surfaces have various descriptions. The three fundamental types are explicit, implicit and parametric description. The following section gives a brief description on these different description types. More information about the geometric fundamentals can be gained in the following literature [2, 4, 6, 8, 11, 12].

2.1.1 Explicit description of curves

This is the simplest description of a curve. Here, one coordinate is described as a function of the other. The general form is $y = f(x)$ and can be further extended to three dimensions. With this description, it is easy to compute derivatives and check whether a given point is on the curve. The major drawback of such curves is that very few curves can be represented in this form and there is a single value of y associated with every x. An explicit description of semi-circle can be written as $y = \sqrt{r^2 - x^2}$, where r is the radius (see Figure 3). Note that only semicircles can be described for each value of x can only correspond to a single value of y. In order to represent a full circle, two independent equations $f_1(x) = \sqrt{r^2 - x^2}$ and $f_2(x) = -\sqrt{r^2 - x^2}$ are needed. Due to the above mentioned reason, the explicit description is seldom used in CAD.

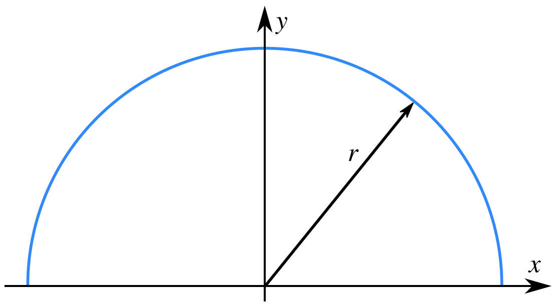


Figure 3: Explicit description of a semicircle

2.1.2 Implicit description of curves

An implicit description of a curve is the set of solutions of an equation of the form $f(x,y) = 0$. For surfaces it is of the form $f(x,y,z) = 0$. It is possible to have multiple values for a given x value, and hence a complete circle can be represented. A circle can be implicitly represented as $x^2 + y^2 - r^2 = 0$, with r being the radius of the circle. With implicit description, in the case of closed curves it is easy to identify if a given point in inside or outside a curve, but determining the intersection of two curves is not straight forward. Implicit descriptions have a limited use in CAD.

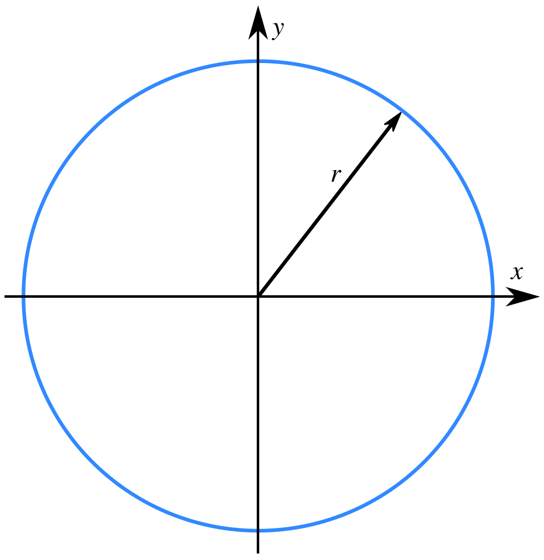


Figure 4: Implicit description of a circle

2.1.3 Parametric description of curves

In this type of description, the coordinates are represented as functions of an independent parameter. This representation is very flexible and a great variety of geometries can be represented. Another advantage is that spatial curves can be represented, whereas using explicit and implicit methods curves can be described only on a plane. The major drawback is that it is difficult to determine if a given point lies on the curve. Figure 5 shows an example for an Archimedean spiral in parametric description.

$$\mathbf{C}(t) = \left\{ \begin{array}{c} x(t) \\ y(t) \end{array} \right\} \text{ with } \begin{array}{l} x(t) = t\cos(\pi t) \\ y(t) = t\sin(\pi t) \end{array} \tag{1}$$

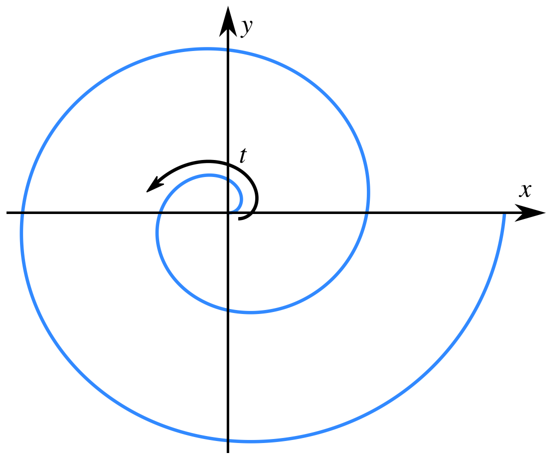


Figure 5: Archimedean spiral represented using paramteric description

2.2 Description of surfaces

Surfaces are 2D entities obtained by the tensor product of two 1D curves. B-Splines and NURBS are the most common description of these 1D curves. The following section discusses B-Spline and NURBS curves briefly.

2.2.1 B-Spline representation of surfaces

Basis-Spline Curves (B-Splines) are defined by a linear combination of control points and basis functions over a parametric space. The parametric domain is divided into intervals and the B-Spline basis functions are defined as piecewise polynomials on these intervals, having certain continuity properties. Since the number of intervals is arbitrary, the choice of the polynomial degree is independent of the number of control points. Figure **??** shows an example of a B-Spline curve.

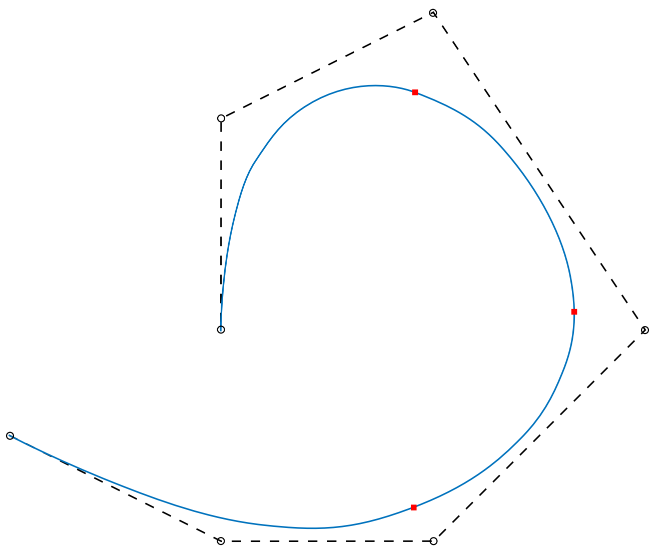


Figure 6: B-Spline curve

Each interval is delimited by knots. Therefore, one can say that the parametric space is defined by a so called knot vector $\Xi = [\xi_1, \xi_2, \ldots, \xi_{n+p+1}]$, where n is the number of control points and p is the polynomial degree. The knot vector is divided into a set of parametric coordinates ξ_i in non-decreasing order, which divide the parametric space into sections. A knot value can appear more than once and is then called a multiple knot. Across a knot with the multiplicity k, the continuity is C^{p-k}. Therefore, by increasing the multiplicity of a knot, the continuity at this knot is reduced. If the first and last knot have a multiplicity of $p+1$, then the knot vector is called an open knot vector. This means that the endpoints are interpolated, instead of being approximated, which makes the curve tangential to the control polygon at these points [4, 12].

Next, the B-Spline basis functions are computed based on the knot vector and the polynomial degree using the Cox-de-Boor recursion formula

$$N_{i,0}(\xi) = \begin{cases} 1\,, & \xi_i < \xi < \xi_{i+1} \\ 0\,, & \text{otherwise} \end{cases} \tag{2}$$

For $p \geq 1$ it is:

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi i + 1} N_{i+1,p-1}(\xi) \tag{3}$$

One of the properties of the B-Spline basis function is that it must be non zero only in the interval $[\xi_i, \xi_{i+p+1}]$. In addition, the partition of unity must be satisfies at all points. Moreover, the basis function have to be non-negative and must be linear independent [4, 6, 12].

A B-Spline curve of degree p is defined by linear combination of control points ($\mathbf{P_i}$) and the respective basis function ($N_{i,p}$) as follows:

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi)\mathbf{P_i}\,. \tag{4}$$

The first derivative is obtained by the linear combination of control points and the derived basis functions ($N'_{i,p}$)as follows:

$$\mathbf{C}'(\xi) = \sum_{i=1}^{n} N'_{i,p}(\xi)\mathbf{P_i} \tag{5}$$

In order to obtain a B-Spline surface, the tensor product of two 1D B-Spline curves is formed as follows:

$$\mathbf{S}(\xi,\eta) = \sum_{i=1}^{n}\sum_{j=1}^{m} = N_{i,p}(\xi)N_{j,q}(\eta)\mathbf{P_{i,j}} \tag{6}$$

Where i and j are the indices in each direction, n and m are the number of control points in each direction, p and q are the polynomials degrees of the respective direction and ξ and η are the variables in the parameteric domain in the respective direction. Figure **??** shows the parametric space of B-Spline surface in 2D.

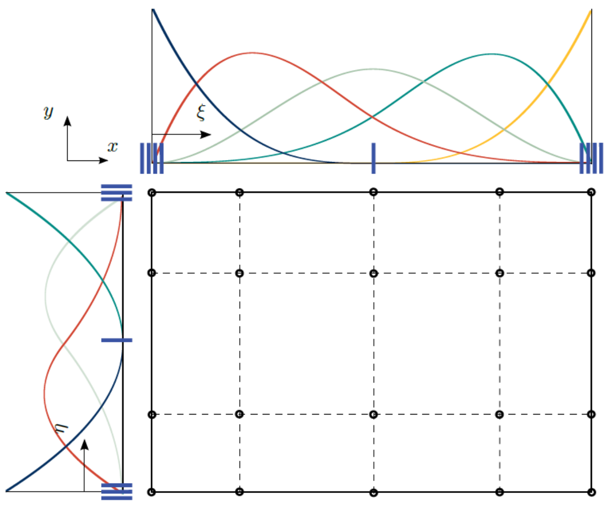


Figure 7: B-Spline surface interpolation [12]

2.2.2 NURBS representation of surfaces

In NURBS, the term Non-Uniform refers to the knot vector, which is not uniform in general and the term rational refers to the basis functions. For NURBS curves, the basis functions are piecewise rational functions [12].

In NURBS curves, each control point has an additional variable assigned to it called weight w_i. Therefore, in order to describe a curve, the following curve equation should be followed:

$$\mathbf{C}(\xi) = \frac{\sum_{i=1}^{n} N_{i,p}(\xi) w_i \mathbf{P_i}}{\sum_{j=1}^{n} N_{j,p}(\xi) w_j} \tag{7}$$

The rational basis functions of NURBS are defined as follows:

$$R_{i,p}(\xi) = \frac{N_{i,p}(\xi) w_i}{\sum_{j=1}^{n} N_{j,p}(\xi) w_j} \tag{8}$$

Therefore, the NURBS curve function can be written as:

$$\mathbf{C}(\xi) = \sum_{i=2}^{n} R_{i,p}(\xi) \mathbf{P_i} \tag{9}$$

If all control point weights are equal, the rational functions in equation (8) reduce to B-Spline functions.

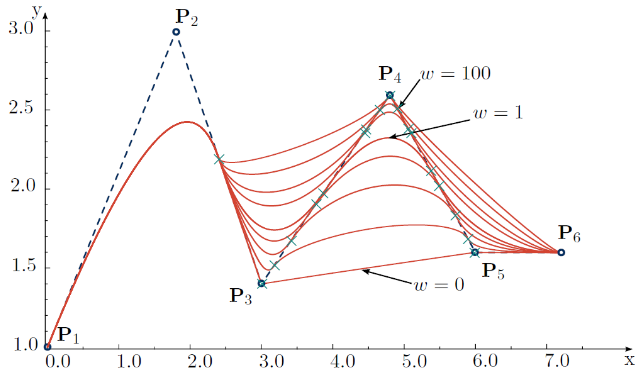


Figure 8: Influence of basis function weight on curve shape [12]

Finally, the NURBS surface basis functions are defined as follows:

$$R_{i,j}(\xi,\eta) = \frac{N_{i,p}(\xi) N_{j,q}(\eta) w_{i,j}}{\sum_{i=1}^{n} \sum_{j=1}^{m} N_{i,p}(\xi) N_{j,q}(\eta) w_{i,j}} \tag{10}$$

The resulting surface is computed as follows:

$$\mathbf{S}(\xi,\eta) = \sum_{i=1}^{n} \sum_{j=1}^{m} R_{i,j}(\xi,\eta) \mathbf{P_{i,j}} \tag{11}$$

In Figure 9, the effect of weight on a control point is seen. As the weight of the control point $\mathbf{P_{3,3}}$ is increased form Figure 9 (a) to Figure 9 (d) the surface is ”pulled” towards the control point.

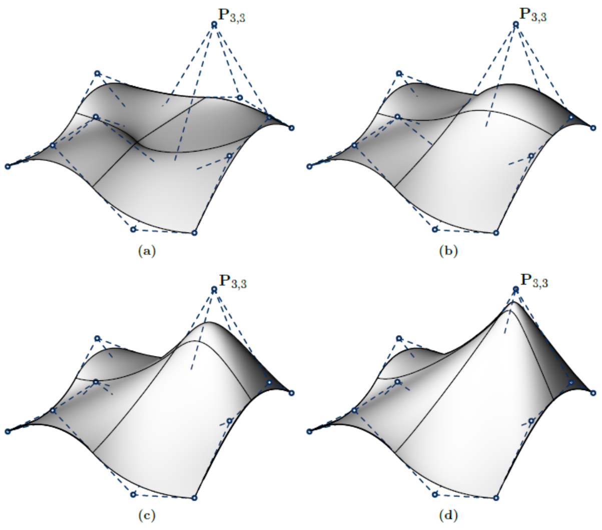


Figure 9: Influence of basis function weight on surface shape [12]

2.2.3 Trimmed NURBS surfaces

Complex surfaces can be effeciently represented using two-dimensional trimmed NURBS. In the present work, a single trimmed NURBS patch is dealt with, which is essentially represented by an untrimmed NURBS surface and a set of trimming NURBS curves. Whereas a surface $\mathbf{S}(\xi,\eta)$ is represented with two parameters ξ and η, a curve is represented by one parameter $\tilde{\xi}$. The trimming curve $\mathbf{C}(\tilde{\xi})$ is a NURBS curve, having control points defined in the parameter space (ξ,η) of the NURBS surface. In CAD, the trimmed surface is basically visualised by showing only the surface which is not cut out by the trimming curves.

As a example, if a CAD model of a plate with a hole inside is needed, the concept of trimming is used. Here, the plate is modelled first and the hole is later represented by a trimming curve. Thus, a trimmed surface is a partially visible surface, where the portion defined by trimming curves is invisible [12].

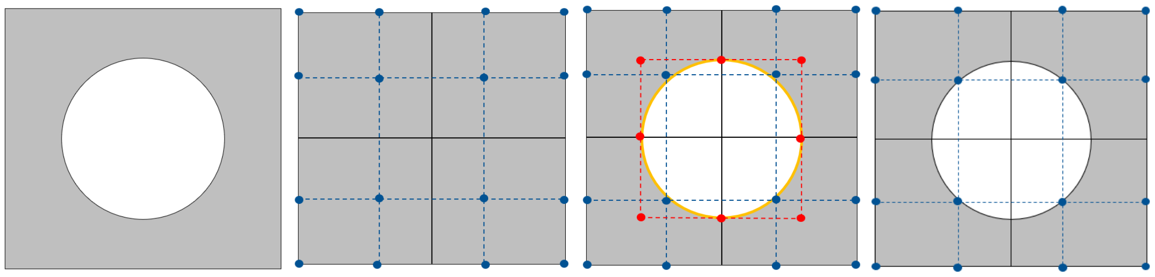


Figure 10: Trimming of a surface

2.3 Isogeometric Analysis

Up until now, complex geometries can be analyzed using FEM, where the domain is decomposed into elements of finite size. These elements are defined by nodal points and connected by basis functions. Often, linear polynomials are used due to their simplicity, however they show no higher than C^0-continuity, which gives a geometric approximation error. In addition, this limitation causes problems in theories like the Kirchhoff-Love shell theory or the Bernoulli beam theory since they require at least C^1-continuity [12].

In CAD B-Reps and NURBS as an implicit geometry description are used to describe surfaces. This is where IGA comes into the modeling, having the goal to incorporate structural analysis with CAD models. This avoids the meshing process by using the aforementioned geometric approximation method for analysis as well. NURBS are the widespread technology in CAD programs and since their basis functions can be used also for analysis, this method has shown advantageous results.

Moreover, it allows analysis without geometry approximation and it shows high continuity which is beneficial for some of the theories mentioned above. In the rest of this section the different components of this method are briefly discussed.

2.3.1 Elements

The current work is limited to working with single NURBS patch. Multiple patches are not considered in this work. The two possible elements definitions. As per the first definition, the elements are defined by the knot vectors in the patch, while a second definition is that, the individual patches are themselves the elements. In the current work, the elements are defined by the knot vectors.

Similar to FEM, NURBS based IGA also have discretization points(nodes), where the degree of freedoms are defined. In IGA, the control points of the geometry are the nodes, where degrees of freedom and boundary conditions are defined. Contrary to classic FEM, the interpolating functions within an element are not confined to one element, as the NURBS basis functions which, are also used are interpolating functions in IGA spread across multiple knots.

2.3.2 Trimming in IGA

Regarding IGA, as shown in Figure 11, there are two different methods for handling a trimmed NURBS surface patch [7]. One method is constructing a new untrimmed multi-patch NURBS surface and implement the analysis upon. Another method is to approximate the trimming curve by polygonizing it and triangulating the intersected elements, as in section 3.2.1, to distribute integration points only in the predefined active region.

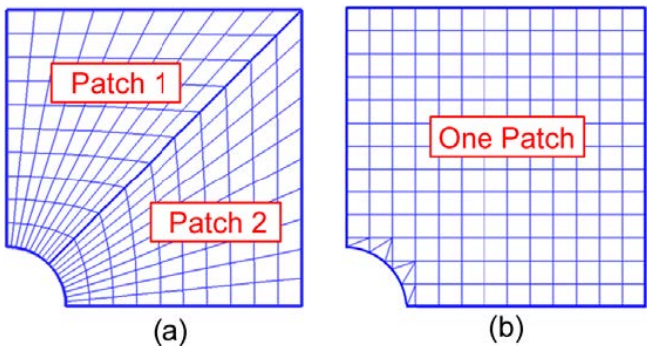
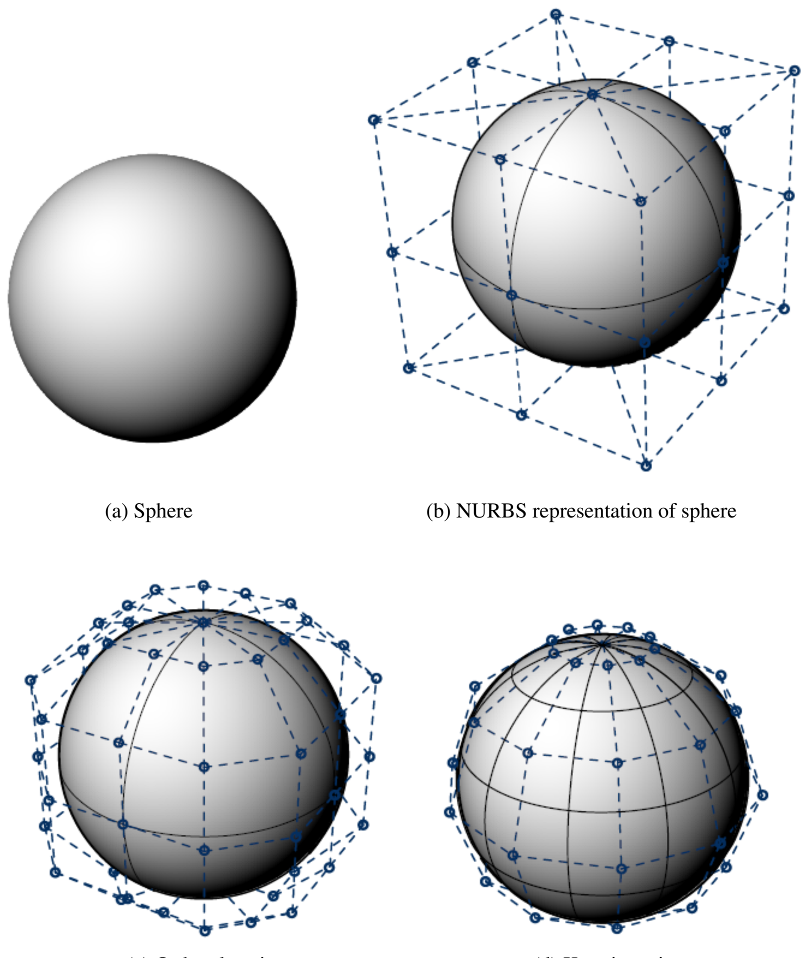


Figure 11: Trimming Methods; (a) Conventional IGA; (b) Trimmed IGA [7]

2.3.3 Refinement

Refinement of the B-Spline entities is an important aspect in IGA. Usually, the description of the geometry uses the minimum amount of control points and knots needed to describe the geometry accurately. This geometry description is not able to properly depict the analysis result and therefore leads to inaccurate results. For this reason the geometry description needs refinement, while the underlying geometry should remain the same. Two basic types are knot insertion and order elevation. Both schemes add additional control points to the geometry, but the final geometry remains unchanged. In knot insertion, which corresponds to h-refinement in regular FEA (Finite Element Analysis), extra knots are added in the parametric space and for each additional knot the continuity is reduced by one. In order elevation, which is similar to the p-extension of the FEA, the order of polynomial is increased, but the number of knot intervals remains unchanged. Increasing polynomial degree results in repeating the existing knots, such that the continuity at the knots remains unchanged. Another type of refinement k-refinement exists, where order elevation is applied first followed by knot insertion [4, 6, 11, 12].


(a) Sphere (b) NURBS representation of sphere

(c) Order elevation (d) Knot insertion

Figure 12: NURBS refinement [12]

2.4 Form finding

For tensile structures, as for example membranes, form finding can be defined as follows:

"Form Finding is the task to find the shape of equilibrium with respect to given surface stress state σ and natural (e.g. edge forces) or geometrical (e.g. clamped edges) boundary conditions" [5]

This means that if a given stress surface σ is applied on the tensile structure, this procedure makes it possible to find the equilibrium shape.

In the following form finding will be explained using a simple example taken from [5]. Take the case of a clothesline hanging between two trees. Whenever a piece of laundry is added, the structure is excited in the beginning until it reaches a balance where no more deformation or displacement occurs and therefore the equilibrium state is reached. This is in general how form finding works.

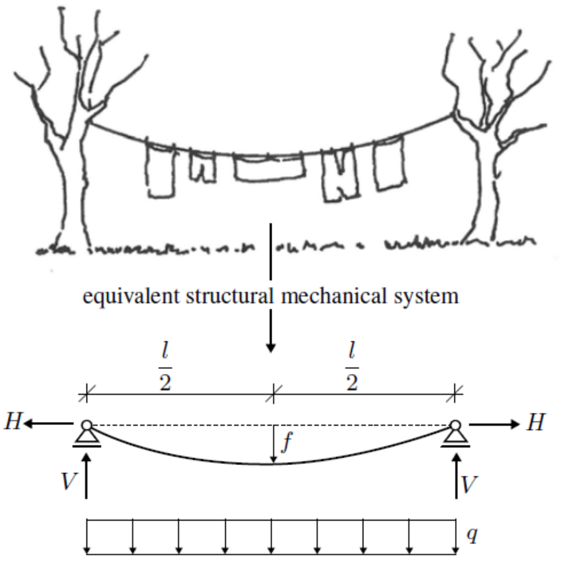


Figure 13: Equilibrium State for Clothesline example [5]

The laundry piece applies a distributed load q that will deform the structure with a displacement f hence applying reaction forces H and V at the supports (both trees) to hold on the structure statically. The displacement f can be then obtained by equilibrium of forces.

Now as a last step, knowing the equilibrium forces n due to this applied load, we can now hold the structure which is the clothesline in this case, by initially applying a prestress H and V that is equivalent to the latter equilibrium state.

The maximum deformation in the middle is directly linked to the amount of applied horizontal forces:

$$f = \frac{q \cdot l^2}{8H} \tag{12}$$

As already described, the main shaping parameter for a tensile structure is the prestress. The relation between the found prestress and geometry can be interpreted from the equilibrium equation, given by:

$$-\delta W_{int} = \int_v \sigma_\mathbf{0} : \nabla \delta \mathbf{u} \, dv = \int_v \sigma_\mathbf{0} : \delta \mathbf{e} \, dv \tag{13}$$

where $\sigma_\mathbf{0}$ is the internal stress, $\delta\mathbf{u}$ is the virtual displacement, $\delta\mathbf{e}$ is the virtual Euler Almansi strain and v is the volume in the current configuration.

$$-\delta W_{ext} = -\int_a \mathbf{n} \cdot \sigma \cdot \delta \mathbf{u} \, da - \int_v \rho \mathbf{b} \cdot \delta \mathbf{u} \, dv \tag{14}$$

where $\mathbf{n}$ is the normal vector to the surface, t is the thickness of the membrane, σ is the external stress applied on the membrane, $\mathbf{b}$ is the body force vector and a is the surface area in the current configuration.

Hence one gets,

$$-\delta W_{tot} = t \int_a \sigma_\mathbf{0} : \nabla \delta \mathbf{u} \, da - \int_a \mathbf{n} \cdot \sigma \delta \mathbf{u} \, da - t \int_a \rho \mathbf{b} \cdot \delta \mathbf{u} \, da = t \int_a \sigma_\mathbf{0} : \nabla \delta \mathbf{u} \, da - \int_a \mathbf{q}_{(\mathbf{n})} \cdot \delta \mathbf{u} \, da \tag{15}$$

where the total virtual work is the sum of both the internal and external virtual work and $\mathbf{q}_{(\mathbf{n})}$ is the external load acting in a certain direction.

In form finding, the final prestress state is first defined and then the equilibrium state is evaluated. Material properties are introduced in a second step when stresses are released to evaluate a cutting pattern.

Since the external stress can be only applied in the current state, Euler Almansi strain will be applied:

$$\begin{aligned} \mathbf{e} &= \frac{1}{2}\frac{l^2 - L^2}{l^2} \qquad \text{in 1-D problems;} \\ \mathbf{e} &= \frac{1}{2}\left(\mathbf{I} - \mathbf{F^{-T}} \cdot \mathbf{F^{-1}}\right) \qquad \text{in 3-D problems} \end{aligned} \tag{16}$$

where l and L are the length of the member in the current and respectively reference configuration, $\mathbf{I}$ is the identity matrix and $\mathbf{F}$ is the deformation gradient.
But one could transfer the integration into the reference state, hence we get:

$$-\delta W = t\int_A \det(\mathbf{F})\sigma_0 : \mathbf{F^{-T}} \cdot \delta\mathbf{E} \cdot \mathbf{F^{-1}}\, dA - \int_a \mathbf{q_{(n)}} \cdot \delta\mathbf{u}\, da = 0 \tag{17}$$

where A is the surface area in the reference configuration and E is the Green-Lagrange strain tensor.

When discretizing and linearizing the equation in space and time, one gets the residual equation that is given by:

$$R_r = t\int_A \det\mathbf{F} \cdot \left(\sigma^{\alpha\beta}\right) \cdot \left(\frac{\partial \mathbf{g}_\alpha}{\partial u_r}\mathbf{g}_\beta\right) dA - \int_a \mathbf{q_{(n)}} \frac{\partial \mathbf{u}}{\partial u_r}\, da = 0 \tag{18}$$

where $\mathbf{g}$ is the base vector in the current configuration.

Applying the Newton-Raphson algorithm, which is a linearized iterative procedure, we need to get the system stiffness matrix:

$$\begin{aligned} K_{rs} = \frac{\partial R_r}{\partial u_s} &= t\int_A \frac{\partial}{\partial u_s}\left(\det\mathbf{F}\left(\sigma_\mathbf{0} \cdot \mathbf{F^{-T}}\right) : \frac{\partial \mathbf{E}}{\partial u_r}\right) dA - \int_a \frac{\partial \mathbf{q_{(n)}}}{\partial u_s} \cdot \frac{\partial \mathbf{u}}{\partial u_r}\, da \\ &= t\int_A \left(\frac{\partial \det\mathbf{F}}{\partial u_s}\sigma^{\alpha\beta}\left(\frac{\partial \mathbf{g}_\alpha}{\partial u_r}\mathbf{g}_\beta\right) + \det\mathbf{F}\frac{\partial \sigma_0^{\alpha\beta}}{\partial u_s}\left(\frac{\partial \mathbf{g}_\alpha}{\partial u_r}\mathbf{g}_\beta\right) + \det\mathbf{F}\sigma^{\alpha\beta}\left(\frac{\partial \mathbf{g}_\alpha}{\partial u_r}\frac{\partial \mathbf{g}_\beta}{\partial u_s}\right)\right) dA \end{aligned} \tag{19}$$

where α and β are 2-D indices.

It turns out that equation (19) is always singular for deformations tangential to the surface, hence equation (18) is unsolvable.

Methods such as the Updated Reference Strategy (URS) are used to stabilize the inverse problem [1].

3 Implementation

As mentioned above the code is implemented in the proprietary software MATLAB. The code reads in a geometry file and implements a solution scheme by firstly enriching the geometry description. Next, all the necessary attributes of the surface patch are stored. Finally, the form finding iterations starts using the aforementioned URS (see section 2.4).

3.1 Preprocessing

The preprocessing stage starts with reading in the geometry. The geometry can be created in a CAD software, such as Rhino, exported as a JSON file, and parsed with MATLAB. The visualization is then accomplished by the means of evaluation points distributed on the parametric space of the surface, which are then transformed to the physical space with the help of both shape functions and control points.

3.1.1 Parsing

To enable the construction of any arbitrary geometry, the code reads in a geometry created by Rhino. A plugin developed by the Chair of Structural Analysis at the Technical University of Munich is used to export a JSON file out of a NURBS surface drawn in Rhino. An example of a complex geometry is depicted in Figure 14. The JSON file contains all the geometrical information necessary to represent the trimmed surface. Its structure first encloses the geometrical data representing the untrimmed surface, which are: the polynomial degree of the untrimmed patch in ξ- and η-directions, the knot vector of the untrimmed patch in ξ- and η-directions, and the control points of the untrimmed Patch. This is shown for the untrimmed four point sail with a hole in Listing 1.

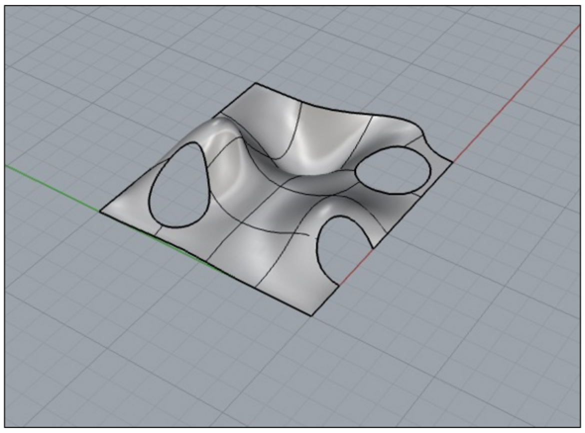
```
{
   "breps": [{
      "brep_id": 1,
      "faces": [{
         "brep_id": 2,
         "swapped_surface_normal": false,
         "surface": {
            "is_trimmed": true,
            "is_rational": false,
            "degrees": [1, 1],
            "knot_vectors": [
               [0, 22.360679774997898],
               [0, 22.360679774997898]
            ],
            "control_points": [
               [1, [10, -10, 10, 1]],
               [2, [10, 10, 0, 1]],
               [3, [-10, -10, 0, 1]],
               [4, [-10, 10, 10, 1]]
            ]
         },
```

Listing 1: JSON snippit for the untrimmed surface

Then, similar information concerning the trimming curve follows, which are: the number of curves in the outer and inner boundary loop, the polynomial degree of the outer and inner trimming curve, the knot vector of the outer and inner trimming curve, the control points of the outer loops defined in counter-clockwise direction, and the control points of the inner loops defined in clockwise direction. An example of a boundary loop is shown in Listing 2.

```
   "boundary_loops": [{
      "loop_type": "outer",
      "trimming_curves": [{
         "trim_index": 0,
         "curve_direction": true,
         "parameter_curve": {
            "is_rational": false,
            "degree": 1,
            "knot_vector": [0, 22.360679774997898],
            "active_range": [0, 22.360679774997898],
            "control_points": [
               [0, 0, 0, 1],
               [22.360679774997898, 0, 0, 1]
            ]
         }
      },
```

Listing 2: JSON snippit for the trimming loops


(a) Geometry Created in Rhino

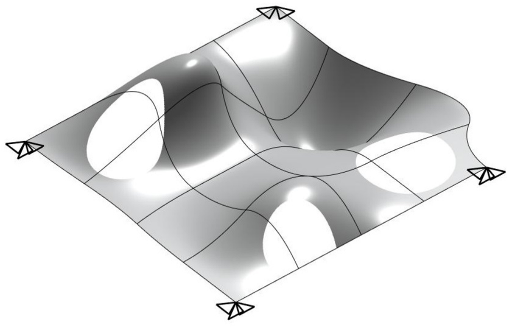

(b) After Parsing and Visualizing in MATLAB

Figure 14: An Example Complex Geometry Parsed from Rhino to MATLAB

3.1.2 Visualization of trimmed geometry with evaluation points

It is desirable to use the Gauss Points (GP) for the visualization of the trimmed surface patch as they are already computed. But unfortunately the usage of the GP for the visualization of a trimmed surface is not possible in the same way it can be done for the visualization of an untrimmed surface. This will be explained later on when introducing triangulation and the distribution of the GP. Due to the complexity of the visualization for trimmed geometries using GP, the evaluations points are used instead.

In this case the following procedure is implemented to show the trimmed geometry:

1. A grid of points is generated in the parametric domain. This can be done by subdividing the span between the first and the last entry of two knot vectors respectively with a certain distance for the new points.

2. For each of the points of this mesh grid, it is tested whether the point lies within the trimmed geometry or not. If not, the parametric coordinate is deleted and a special data type value representing an undefined value (NaN) is inserted.

3. For each of the points of this mesh grid the coordinates in the cartesian domain X_p, Y_p and Z_p are calculated. If the coordinate value in parametric domain is a NaN, also the coordinate values in the cartesian domain is a NaN.

4. The *surf* command in MATLAB using the cartesian coordinates X_p, Y_p and Z_p shows the surface interpolated by these points.

The undeformed trimmed geometry using evaluation points for the visualization can be seen in Figure 15.

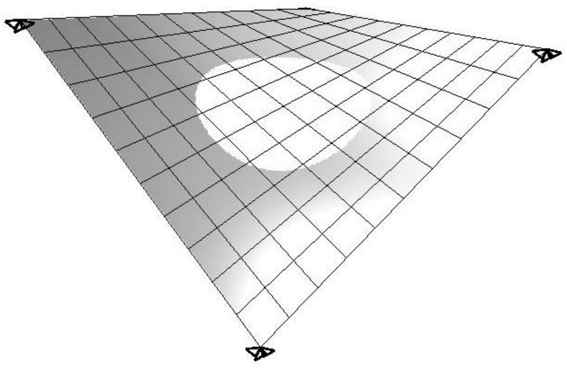


Figure 15: Visualization of trimmed geometry before form finding

3.2 Structural analysis

3.2.1 Triangulation and distribution of Gauss points

Usually, the weak form cannot be integrated analytically. To overcome this problem, numerical integration schemes, like the Gauss-Legendre Quadrature, are applied. By evaluating the integrand at discrete weighted points and summing up these values, the analytical integration is replaced [3, 4, 12].

$$\int_{xo}^{x1} f(x)\,dx = \sum_{i=1}^{n} f(x_i)\cdot w_i \cdot \det(\mathbf{J}) \tag{20}$$

If polynomials are used as test functions, the Gauss quadrature can yield the exact integral value if the number of quadrature points is above the required number of GPs ($n_{req} = (p+1)/2$). However, in IGA usually NURBS are used as ansatz functions, which are not polynomial, but rational functions. Still the fact holds, that the Gauss Quadrature delivers adequately accurate results if enough points are being used ($n_{req} = p+1$) [12].

Since all the computations on element-level are being done on the GPs, their distribution is one of the most important aspects in the analysis process. The distribution can be accomplished in two ways:

1. Distribute GPs in the untrimmed patch and delete all the GPs, which are inside the trimming polygon.
2. Trim the patch and find all the elements that are being intersected by the trimming curve. Triangulate the intersected elements and distribute GPs in the triangulated patch.

The first option has the advantage of a simple implementation. However, it also has the major disadvantage that heavily trimmed elements might end up with no Gauss Points and therefore zero contribution to the global stiffness matrix. For this reason the second option was chosen. The process of triangulating and distributing GPs with the correct Gauss rule can be divided into four steps, which will be explained in the following.

With the help of MATLAB's Mapping Toolbox, elements can be formed using the built-in function *polyshape*. *Polyshape* creates a polygon defined by 2d vertices and returns an object with properties describing its vertices, solid regions and holes.

The surface patch and the trimming curves are defined in separate spaces. These individual spaces are connected in the following manner: The physical space of the trimming curve corresponds to the parametric space of the surface. Therefore the trimming curves have to be mapped into their physical space. Therefore, elements intersected by the boundary loops can be found easily using operations on sets such as *intersection* and *subtraction of polyshapes*. The algorithm is shown on the example of the four point sail with a hole in the middle as shown in Figure 16.

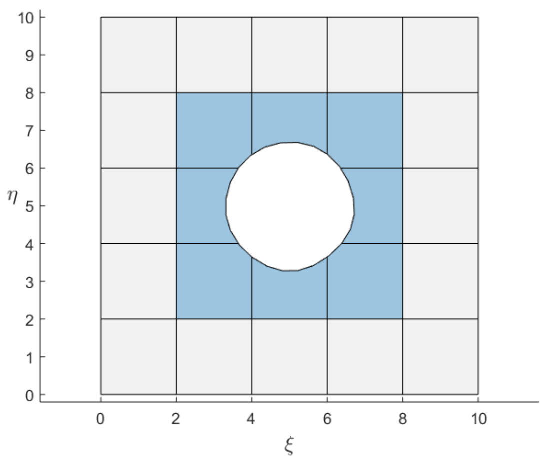


Figure 16: Elements intersected by the Trimming Curve

By using the set operations *intersect* and *subtract*, elements affected by the boundary loop can be found. These elements correspond to the blue elements in Figure 16. As mentioned above these elements have to be triangulated.

However, the hole should not be penetrated by the triangles. For this reason, a constrained Delaunay Triangulation is to be used [9]. This method ensures that not only the boundary is going to be respected, but also concavities are not filled with triangles. Since polyshape objects are used, they can be triangulated using the built-in triangulation for *polyshape* command. Therefore, the bounding edges don't need do be defined explicitly. The resulting triangulated four point sail can be seen in Figure 17.

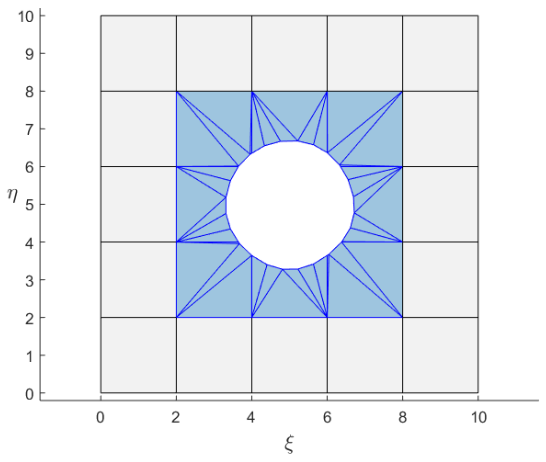


Figure 17: Triangulation of intersected Elements

Finally, the elements have to be filled with GPs. Elements that are not intersected by the trimming curve (marked in grey) have to be filled with a Gauss rule for quadrilaterals. The GPs have to be mapped from the canonical space, which is spanned in the interval [-1,1], to the parameter space [3, 4, 12]. The linear mapping of the Gauss points is done using the following equations:

$$\xi = \frac{(\Xi_{i+1} + \Xi_i) + \xi_{GP_i}\,(\Xi_{i+1} - \Xi_i)}{2}\,, \tag{21}$$

$$\eta = \frac{(H_{i+1} + H_i) + \eta_{GP_i}\,(H_{i+1} - H_i)}{2}\,. \tag{22}$$

Using this approach yields the GPs in the parameter domain, which is shown in Figure 18a. Similarly the Gauss points of the triangulated elements can be mapped from the canonical triangle, which is also defined in the interval [-1,1], to the parameter domain. Since the mapping is linear, the regular triangular shape functions can be used:

$$\xi = N_1\,x_1 + N_2\,x_2 + N_3\,x_3 = \sum_{i=1}^{n} N_i\,x_i\,, \tag{23}$$

$$\eta = N_1\,y_1 + N_2\,y_2 + N_3\,y_3 = \sum_{i=1}^{n} N_i\,y_i\,, \tag{24}$$

where

$$\begin{aligned} N_1 &= 1 - \xi - \eta\,, \\ N_2 &= \xi\,, \\ N_3 &= \eta\,. \end{aligned} \tag{25}$$

The distribution of the GPs in the triangular elements are depicted in red in Figure 18b.

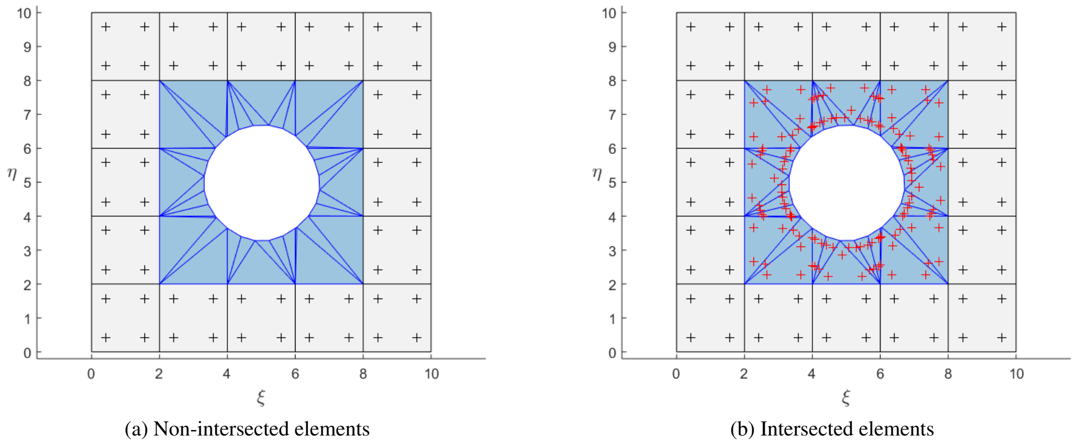


Figure 18: Filling elements with Gauss Points

The number of GPs in the not intersected elements is known a priori, since it depends on the polynomial degree, as mentioned above. The number of GPs in intersected elements however, can only be known a posteriori. It depends on the number of triangles triangulating the intersected element. Thus, the number of GPs varies elementwise. The GPs for the whole structure therefore cannot be stored in a preallocated matrix, since the number of GPs is not known beforehand. Furthermore, the GPs no longer are positioned grid-wise. This on the other hand is required by the *surf* command, which is able to produce a smooth surface. Other MATLAB commands that can deal with scattered data do not produce surfaces as smooth as these. The saving of computational effort can therefore not justify the usage of GPs for visualization and evaluation points are used for visualization instead (see also section 3.1.2).

3.2.2 Cables

For the membrane theory to be applicable, extra stiffness is needed to be added on the boundaries where cables are embedded. This is implemented on the above mentioned example of the four point sail with a circular hole in middle as shown in Figure 19.

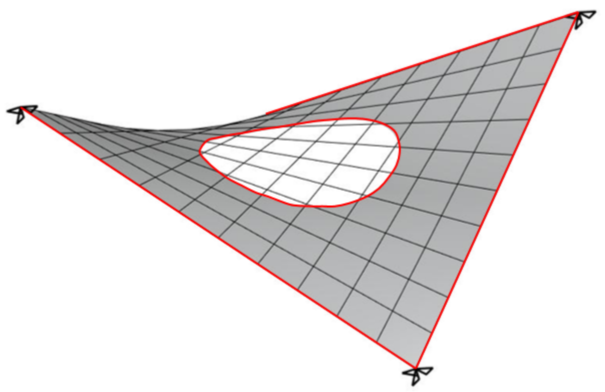


Figure 19: Boundaries of the trimmed surface

To apply stiffness on the ”cables,” essentially, the internal virtual work of the boundary is used, which is:

$$\delta W_{int} = -\int_{\Omega} (\mathbf{n} + \mathbf{n_0}) : \delta \mathbf{e} \, d\Omega \, , \tag{26}$$

where $\mathbf{n}$ is the normal stress, $\mathbf{n_0}$ is the normal prestress, and $\delta\mathbf{e}$ is the variation in the strain. Using the equilibrium condition of $\delta W = 0$, the equilibrium must be also applied with respect to the virtual displacement field $\delta\mathbf{u}$, as depicted in:

$$\delta W = \frac{\partial W}{\partial \mathbf{u}} \delta \mathbf{u} = 0 \, . \tag{27}$$

The discretized version of equation (27) is:

$$\delta W = -\mathbf{R} \cdot \delta \mathbf{u_h} = 0 \, , \tag{28}$$

where $\mathbf{R}$ is the residual vector and $\delta \mathbf{u_h}$ is the variation of the approximate displacement field. The tangent stiffness matrix K_{ij} is defined as the following:

$$K_{ij} = \frac{\partial R_i}{\partial u_j} = -\frac{\partial^2 W}{\partial u_i \partial u_j} \, . \tag{29}$$

To achieve the stiffness due to the prestress as in equation (29), the tangent vector to the trimming curve in the physical space of the surface $\mathbf{T_2}$ is needed. It is calculated as the following:

$$\mathbf{T_2} = \frac{\partial \mathbf{X}_{surf}}{\tilde{\theta}} = \frac{\partial \mathbf{X}_{surf}}{\theta^1} \frac{\theta^1}{\tilde{\theta}} + \frac{\partial \mathbf{X}_{surf}}{\theta^2} \frac{\theta^2}{\tilde{\theta}} = \mathbf{A_1} \hat{T}_1 + \mathbf{A_2} \hat{T}_2 \, , \tag{30}$$

where $\mathbf{X}_{surf}$ is the parametric surface equation, $\tilde{\theta}$ is the parameter defining the trimming curve, θ^1 and θ^2 are the parameters defining the surface, $\mathbf{A_1}$ and $\mathbf{A_2}$ are the tangent vectors to the surface in the physical space of the surface, and $\hat{T}_1$ and $\hat{T}_2$ are the tangent vectors to the trimming curve in its parametric space (which is the parametric space of the surface). These parameters are illustrated in Figure 20.

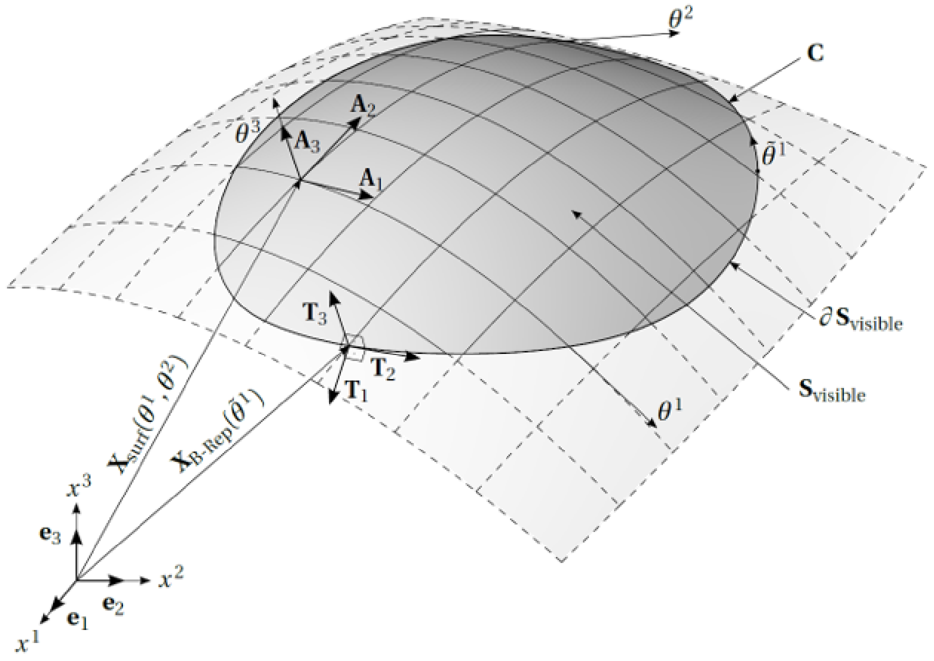


Figure 20: Tangent Vectors of the Trimming Curves [2]

3.2.3 Tangent stiffness after trimming

Since the control points, which affect only the area in the trimmed out part of the geometry, still exist, these control points have no contribution to the stiffness of the system. Moreover, the integration points that happen to lie inside the trimmed region have a tributary area of zero, which leads to some NaN values throughout the form finding process. Thus, at the point of dividing by zero, all the resulting NaN values have to be changed to zeros, since they would have no contribution.

Also, the control points which have no contribution to the stiffness, would fly freely during the form finding process (as depicted in Figure 22a). The remedy to this is to find the rows in the tangent stiffness matrix which have zero values (with some tolerance) and assign a homogeneous Dirichlet boundary condition to the respective degree of freedom.

The figures below illustrate the resulting geometry after one form finding steps, with and without assigning the homogeneous boundary conditions.

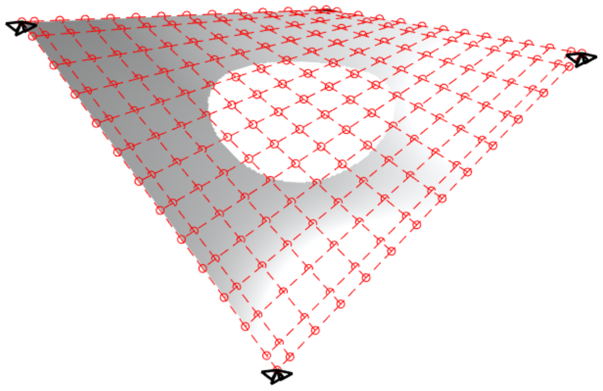
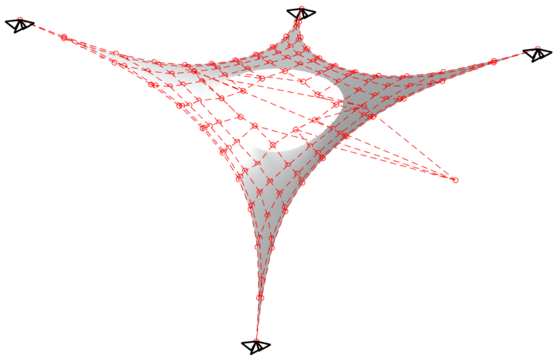


Figure 21: Four Point Sail before form finding


(a) Without adding homogeneous boundary conditions

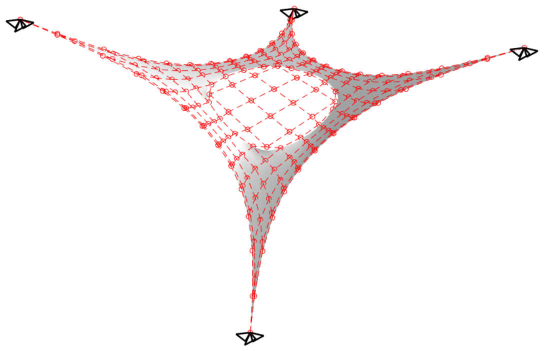
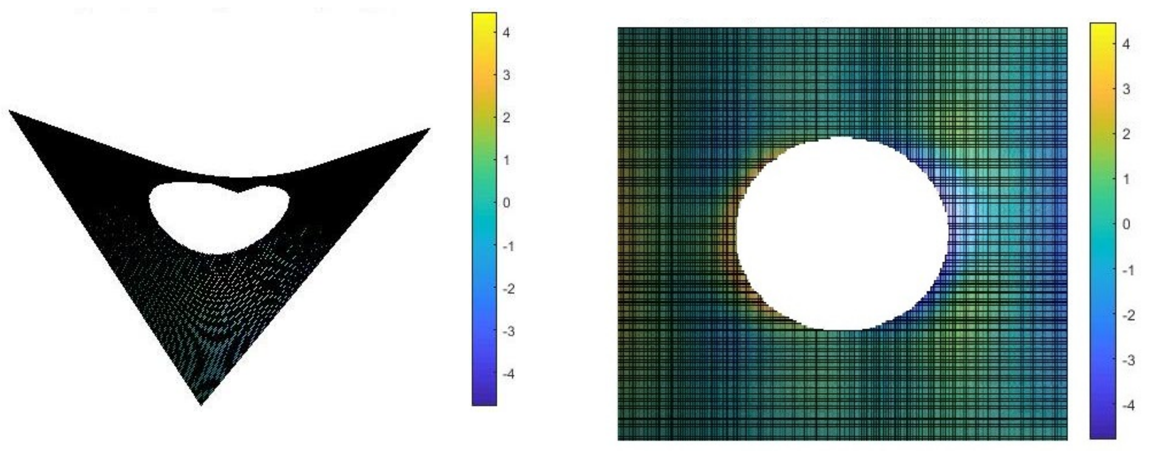

(b) With adding homogeneous boundary conditions

Figure 22: Four point sail with a hole after one step of form finding

3.3 Postprocessing

3.3.1 Visualization of the displacement during the form finding process

Regarding the visualization of the geometry before the form finding process, the same can be done for showing the displacement during the form finding process. During the form finding process the positions of the Control points are changed due to the requirements of equilibrium in form finding. The new Control points are stored and can again be used to computed the cartesian coordinates X_p, Y_p and Z_p in step 3 of the procedure described in section 3.1.2.



(a) 3D view

(b) planar view

Figure 23: Shift of evaluation points in x-direction after form finding

To show the shift of each evaluation point during the form finding process, the shift in each iteration step has to be summed up. Now for each evaluation point three values of how much the point was shifted in each of the three cartesian coordinate directions x, y and z are obtained. Figure 23 shows how much each evaluation point is shifted in x-direction (here the horizontal direction) during the form finding process. The intersection points of the black lines show the positions of the evaluation points. It can be seen that here the boundaries at the inner trimming curve move inwards in a way that the hole disappears whereas the outer trimming curves are fixed at the four corners but move inwards a bit as well.

A trimmed surface is a surface where only the untrimmed parts are visible, but the trimmed regions are geometrically still there. So when computing the results (e.g. the displacements) for the evaluation points, also results for the trimmed out regions are determined. In order not to show the values for the trimmed regions, the matrices containing the results have to be modified by inserting NaNs in all those positions where the matrices of X_p, Y_p and Z_p contain NaNs.

4 Conclusions and Outlook

After implementing the JSON parser to read in geometry information directly from Rhino, the extended MATLAB code is now able to visualize the trimmed NURBS Surface. Then, to find the intersected elements, the trimming curves were linearized and by a boolean operation, it could be defined which elements are intersected by the trimming curves. For the untrimmed elements, the integration points were distributed based on the quad rule. However for the intersected elements, a triangulation is performed in order to distribute integration points inside the corresponding structure.

For the example of the four point sail with a circular hole in the middle, the structure behaved differently based on the prestress of the cables. When the prestress of the cables was relatively large, the form finding iterations converged to a state where the hole is closed. On the other hand, when the prestress of the cables was relatively small, the membrane structure reached a state where the membrane diminished and the cables were connected at the midpoint.

In the post processing task, even if the results of the form finding were not as expected and were diverging for some parameters for the trimmed structure, the visualization of the form finding process was established.

In future work, the influence of the prestress of the cables on the form finding process has to be further evaluated. Furthermore, it has to be worked out how a stable and converging form finding process can be implemented for a trimmed NURBS surface patch. The current work is also limited to the usage of a single patch and can be extended to multi patch NURBS surfaces. Also, Using the Gauss points of the elements could be used as an efficient way to visualize the geometry. Therefore, further work on a better distribution of these Gauss points across a trimmed surface could be implemented.

References

[1] K.-U. Bletzinger and E Ramm.
"A General Finite Element Approach to the form Finding of Tensile Structures by the Updated Reference Strategy".
In: *International Journal of Space Structures* 14.2 (1999), pp. 131–145.
DOI: `10.1260/0266351991494759`.

[2] M. Breitenberger et al.
"Analysis in computer aided design: Nonlinear isogeometric B-Rep analysis of shell structures".
In: *Computer Methods in Applied Mechanics and Engineering* 284 (2015), pp. 401–457.
DOI: `10.1016/j.cma.2014.09.033`.

[3] I. N. Bronstein et al.
Taschenbuch der Mathematik.
Harri Deutsch, 2012.
ISBN: 3817120184.

[4] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs.
Isogeometric Analysis: Toward Integration of CAD and FEA.
Wiley, 2009.
ISBN: 9780470748732.

[5] F. H. Dieringer.
"Numerical Methods for the Design and Analysis of Tensile Structures".
PhD thesis. Technische Universität München, 2014.

[6] J. Kiendl.
"Isogeometric Analysis and Shape Optimal Design of Shell Structures".
PhD thesis. Technische Universität München, Mar. 2011.

[7] H.-J. Kim, Y.-D. Seo, and S.-K. Youn.
"Isogeometric analysis for trimmed CAD surfaces".
In: *Computer Methods in Applied Mechanics and Engineering* 198.37 (2009), pp. 2982 –2995.
DOI: `https://doi.org/10.1016/j.cma.2009.05.004`.

[8] S. Kollmannsberger, B. Wassermann, and E. Rank.
"Computation in Engineering I : Geometric Modeling".
Lecuture Notes. Technische Universität München, 2016.

[9] MathWorks.
Delaunay Triangulation.
URL: `https://de.mathworks.com/help/matlab/math/delaunay-triangulation.html#bspqi8a-12`.

[10] B. Philipp et al.
"Integrated design and analysis of structural membranes using the Isogeometric B-Rep Analysis".
In: *Computer Methods in Applied Mechanics and Engineering* 303 (May 2016), pp. 312–340.
DOI: `10.1016/j.cma.2016.02.003`.

[11] L. Piegl and W. Tiller.
The NURBS Book.
Springer-Verlag, 1996.
ISBN: 9783540615453.

[12] R. Wüchner, M. Breitenberger, and A. Bauer.
"Isogeometric Structural Analysis and Design".
Lecuture Notes. Technische Universität München, 2016.