49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition
4 - 7 January 2011, Orlando, Florida

AIAA 2011-891

49th AIAA Aerospace Sciences Meeting, January 4–7, 2011, Orlando, FL USA

An advanced NURBS fitting procedure for post-processing of grid-based shape optimizations

Gerrit Becker [*] and Michael Schäfer [†]
Technische Universität Darmstadt, Dolivostr. 15, 64293 Darmstadt, Germany

Antony Jameson [‡]
Stanford University, Stanford, 94305, California, USA

An advanced NURBS fitting procedure for shape optimizers in 2D as well as 3D is presented. This approach utilizes a two step fitting process to approximate given sets of structured and unstructured grid points with NURBS curves or surfaces. This back end can be used for instance as post processor for Jameson's well known shape optimization codes SYN83 and SYN107. Hence, the proposed procedure allows an easy transfer of the optimized, grid-based shapes into, for CAD users easy to implement NURBS curves or surfaces.

I. Introduction

Nowadays shape optimization of grid-based aerodynamic simulations has been performed for several years in academia as well as industry. Most of these approaches use a computational grid to alter the wing's shape, e.g Jameson's SYN83 and SYN107 (see Ref. 1). Hence, the optimized grid point positions are usually the final output. In most industrial applications, however CAD software is used to represent shapes with high-order surface descriptions, e.g. NURBS surfaces. Thus, a demand for an accurate and easy to use post processor which translates a given set of grid points into a NURBS surface representation has arisen.

Therefore, within this paper we present a new and advanced two-stage NURBS surface fitting scheme to approximate given surface points with high accuracy and minimal deviations. The approach is based on the work of Ref. 2–6 and proposes an improved fitting scheme which includes the optimization of the NURBS weight values, which even allows exact circular shape representations. In addition to an approach in 2D and for structured grids in 3D, a new and advanced fitting scheme for unstructured grids in 3D is presented.

II. About NURBS

In this section we like to present a brief overview of nonuniform rational B-spline (NURBS) curves and surfaces which are extensively used in the CAD world. Therefore, a short introduction about splines and the derivation of NURBS surfaces in $\mathbb{R}^3$ with a regular parameterization is given. However, later in this section we need to distinguish between the derivation of NURBS curves and NURBS surfaces for structured as well as unstructured grids.

A. Background

According to Piegl in Ref. 7, NURBS curves are vector-valued piecewise rational polynomial functions (see Ref. 8–10 for further reference). NURBS are used within this work to calculate an approximation of given surface grid points which represent the optimized shape. This is a powerful technique to change the grid

[*]Doctoral Candidate, Institute for Numerical Methods in Mechanical Engineering, Technische Universität Darmstadt and Department of Aeronautics and Astronautics, Stanford University, AIAA Student Member.
[†]Professor, Institute for Numerical Methods in Mechanical Engineering, Technische Universität Darmstadt.
[‡]Professor, Department of Aeronautics and Astronautics, Stanford University, AIAA Fellow.

point position and therefore the shape of the objects by adjusting the control points. In contrast to the amount of grid points the amount of control points is rather small. Starting from splines the definition of NURBS surfaces is shown step by step.

Later a linear regression problem is solved in order to fit a NURBS surface to the provided surface grid points and gives us a first set of control points.

B. B-spline-surfaces

B-splines are constructed using a particular class of polynomial functions. These functions are called B-spline basis functions.[10]

The finite polynomial B-spline surface $S : [0,1] \times [0,1] \to \mathbb{R}^3$ is a product of two piecewise defined polynomial functions of order p, q as well as $\hat{n}$,$\hat{m}$ denote the number of supporting points in each direction. Therefore, we define two knot vectors $\xi = (\xi_1, \xi_2, \dots, \xi_{\hat{n}+p})$ and $\nu = (\nu_1, \nu_2, \dots, \nu_{\hat{m}+q})$ by

$$\xi_i = \begin{cases} 0 & \text{if} \quad i < p \\ \frac{i-p}{\hat{n}-p} & \text{if} \quad p \le i \le \hat{n} \\ 1 & \text{if} \quad i > \hat{n} \end{cases} \qquad \text{for} \qquad i = 1, \dots, \hat{n}+p \tag{1}$$

and ν analogously.

According to Ref. 11 the above definition generates a uniform knot vector, which leads to a *uniform* B-spline surface. Furthermore, the end knots are repeated p (resp. q) times. As a consequence the surface's corner points coincide with the corresponding control points. This allows a direct manipulation. A nonuniform knot vector would lead to a *nonuniform* B-spline surface. Section V will show details about the nonuniform knot vector computation used in our context.

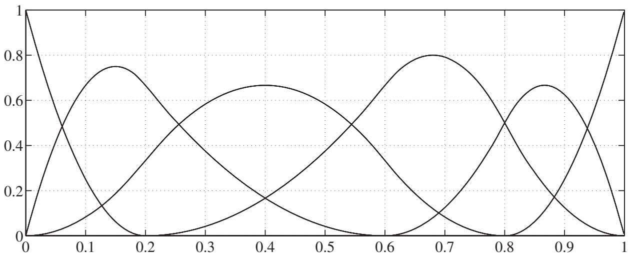

Figure 1. B-spline basis functions $N^{p=3}_{i\in\{1,\dots,6\}}$ using the nonuniform knot vector $\xi = (0, 0, 0, 0.2, 0.6, 0.8, 1, 1, 1)$.

The B-Spline basis functions $N^p, N^q : [0,1] \to \mathbb{R}$, with $N_i^p := N^p_{|[\xi_i,\xi_{i+1}]}$ (N_j^q using ν analogously) are computed piece wisely and recursively with the Cox-deBoor recurrence[12] (see also Fig. 1). Note, that we follow the convention of Farin et al. (Ref. 13): $0/0 := 0$.

$$N_i^1(u) = \begin{cases} 1 & \text{if } \xi_i \le u < \xi_{i+1} \\ 0 & \text{otherwise,} \end{cases}$$

$$N_i^p(u) = \frac{u - \xi_i}{\xi_{i+p} - \xi_i} N_i^{p-1}(u) + \frac{\xi_{i+p+1} - u}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1}^{p-1}(u), \qquad (N_j^q(v) \text{ analogously}). \tag{2}$$

Therefore, following Piegl and Tiller in Ref. 14,

$$S(u,v) = \sum_{i=1}^{\hat{n}} \sum_{j=1}^{\hat{m}} N_i^p(u) N_j^q(v) P_{i,j}, \quad u, v \in [0,1], \tag{3}$$

where $P_{i,j} \in \mathbb{R}^3$ denotes the (i,j)th entry of the control points $P \in \mathbb{R}^{\hat{n}} \times \mathbb{R}^{\hat{m}} \times \mathbb{R}^3$.

C. Nonuniform rational B-spline surfaces (NURBS surfaces)

A NURBS surface is a B-spline surface with a nonuniform knot vector[10] and is defined as follows:

$$S(u,v) = \frac{\sum_{i=1}^{\hat{n}} \sum_{j=1}^{\hat{m}} w_{i,j} P_{i,j} N_i^p(u) N_j^q(v)}{\sum_{i=1}^{\hat{n}} \sum_{j=1}^{\hat{m}} w_{i,j} N_i^p(u) N_j^q(v)}, \quad u,v \in [0,1] \tag{4}$$

with weights $w = (w_{i,j})_{i=1,\ldots,\hat{n};j=1,\ldots,\hat{m}}$. The computation of the nonuniform knot vectors ξ, ν will be discussed in Section V.

III. Remarks about discrete NURBS computation

After defining general NURBS surfaces, this section presents the derivation of discrete NURBS descriptions and their notation in $\mathbb{R}^2$ and $\mathbb{R}^3$ which will be used throughout this paper. For the proposed reverse engineering process (i.e. the computation of the control point locations based on given grid points) the parameterization and the knot vector generation is essential. Thus, in Sections IV and V we present detailed algorithms for the parameterization and knot vector computation of three grid types (2-d, 3-d quadrilateral structured and 3-d triangular unstructured).

However, we will first present the notation of discrete NURBS curves and surfaces. Please note, that the literature usually uses a matrix representation of the grid points and control points in 3-d as you can see in Eq. (3) and Eq. (4). For our purpose it is more useful, especially for unstructured grids, to use a list of grid points and control points as we naturally use in 2-d.

A. Notation in $\mathbb{R}^2$

The curve points

$$X = (X_1, \ldots, X_n), \text{ where } X_i \in \mathbb{R}^2, \ i \in \{1, \ldots, n\} \tag{5}$$

are represented by a set of control points

$$P = (P_1, \ldots, P_{\hat{n}}), \text{ where } P_j \in \mathbb{R}^2, \ j \in \{1, \ldots, \hat{n}\}, \tag{6}$$

with their respective weights $w = (w_j)_{j=1,\ldots,\hat{n}}$ and the B-Spline basis functions of order p

$$N^p = (N_{i,j}^p)_{i=1,\ldots,n;j=1,\ldots,\hat{n}}. \tag{7}$$

In addition we need a suitable parameterization

$$u = (u_1, \ldots, u_n), \text{ where } u_i \in [0,1], \ i \in \{1, \ldots, n\} \tag{8}$$

as well as a knot vector $\xi = (\xi_1, \ldots, \xi_{\hat{n}+p})$ to compute N^p using Eq. (2). The automatic generation of u and ξ based on a given X is presented in Sections IV and V. Applying the above leads to the to following NURBS equation in 2-d

$$X_i = \frac{\sum_{j=1}^{\hat{n}} P_j w_j N_{i,j}^p}{\sum_{j=1}^{\hat{n}} w_j N_{i,j}^p}, \quad i \in \{1, \ldots, n\}. \tag{9}$$

B. Notation in $\mathbb{R}^3$

In 3-d we distinguish between structured and unstructured grids, but we can use the same NURBS equation for computation. However, please note that the parameterization and knot vector generation has to be handled differently. This will be presented in Sections IV and V. Let

$$X = (X_1, \ldots, X_n), \text{ where } X_i \in \mathbb{R}^3, \ i \in \{1, \ldots, n\} \tag{10}$$

be a list of grid points, which is represented by a list of control points of length $\hat{k} = \hat{n}\hat{m}$:

$$P = (P_1, \ldots, P_{\hat{n}\hat{m}}), \text{ where } P_j \in \mathbb{R}^3, \ j \in \{1, \ldots, \hat{k}\}. \tag{11}$$

$\hat{n}$ denotes the number of control points in the first direction and $\hat{m}$ in the second direction. Furthermore, we define $w = (w_j)_{j=1,\ldots,\hat{k}}$ to be their respective weights and the B-Spline basis functions in each direction of order p and q

$$N^p = (N^p_{i,j})_{i=1,\ldots,n;j=1,\ldots,\hat{n}} \tag{12}$$
$$N^q = (N^q_{i,k})_{i=1,\ldots,n;k=1,\ldots,\hat{m}}. \tag{13}$$

To be able to solve the matrix based Eq. (4) we combine N^p and N^q to one term

$$N_{i,(j-1)\hat{m}+k} = N^p_{i,j} N^q_{i,k}, \ \forall i, \ j \in \{1,\ldots,\hat{n}\}, \ k \in \{1,\ldots,\hat{m}\}. \tag{14}$$

Furthermore, let

$$u = (u_1,\ldots,u_n), \text{ where } u_i \in [0,1], \ i \in \{1,\ldots,n\} \text{ and} \tag{15}$$
$$v = (v_1,\ldots,v_n), \text{ where } v_i \in [0,1], \ i \in \{1,\ldots,n\} \tag{16}$$

be a suitable parameterization and let $\xi = (\xi_1,\ldots,\xi_{\hat{n}+p})$, $\nu = (\nu_1,\ldots,\nu_{\hat{m}+q})$ be the knot vectors in both directions. This allows the computation of the basis functions N^p and N^q according to equation (2). Applying the above yields to the following equation

$$X_i = \frac{\sum_{j=1}^{\hat{k}} P_j w_j N_{i,j}}{\sum_{j=1}^{\hat{k}} w_j N_{i,j}}, \quad i \in \{1,\ldots,n\}. \tag{17}$$

IV. Parameterization of given grid points

This section presents parameterization approaches based on the grid type. The basis function computation using the Cox-deBoor[12] recurrence relies on a good parameterization technique as these parameters are the mapping to the real coordinates.

A. 2-d case

The 2-d case is very straight forward, thus we propose the following parameterization procedure based on Ref. 2,4,5. Let $X^0 = (X^0_1,\ldots,X^0_n), X^0_i \in \mathbb{R}^3$, $i \in \{1,\ldots,n\}$ be a set of given grid points, we compute parameter

$$u_1 = 0, \quad u_i = \frac{\sum\limits_{j=1}^{i-1} \left(\|X^0_{j+1} - X^0_j\|\right)^e}{\sum\limits_{j=1}^{n-1} \left(\|X^0_{j+1} - X^0_j\|\right)^e}, \quad i = 2,\ldots,n \tag{18}$$

where $e \in [0,1]$ introduced by Ref. 5 denotes the parameterization type. $e = 0$ leads to an uniform distribution of u, whereas $e = 1$ leads to a chord length distribution of the parameterized curve points X^0. Furthermore, $e = 0.5$ leads to a centripetal distribution. We suggest the chord length distribution $e = 1$ as it serves best our needs to map the real curve coordinates from $\mathbb{R}^2$ to $[0,1]$ (see also Fig. 2).

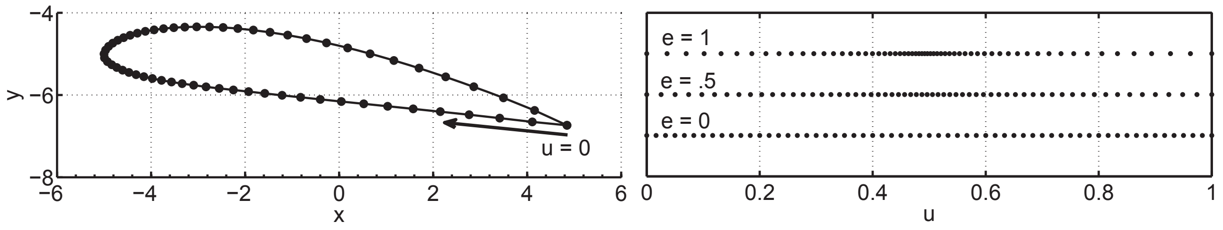

Figure 2. NACA 4415 wing in 2-d and its parameterization with different parameterization types

B. 3-d case with structured grids

For parameterizing structured quadrilateral surface meshes we propose an extended method based on Eq. (18) and the work of Ma and Kruth in Ref. 4. Let $X^0 = (X_1^0, \dots, X_n^0), X_i^0 \in \mathbb{R}^3,\ i \in \{1, \dots, n\}$ be again a set of given grid points and let $\tilde{n}, \tilde{m}$ be the number of grid points in the first and second surface direction, where $n = \tilde{n}\tilde{m}$. Thus we compute parameters

$$u_{(i-1)\tilde{n}+1} = 0, \quad u_{(i-1)\tilde{n}+j} = \frac{\sum_{k=1}^{j-1} \left(\|X^0_{(i-1)\tilde{n}+k+1} - X^0_{(i-1)\tilde{n}+k}\| \right)^e}{\sum_{k=1}^{\tilde{n}-1} \left(\|X^0_{(i-1)\tilde{n}+k+1} - X^0_{(i-1)\tilde{n}+k}\| \right)^e}, \quad i \in \{1, \dots, \tilde{m}\},\ j \in \{2, \dots, \tilde{n}\} \tag{19}$$

and

$$v_j = 0, \quad v_{i\tilde{n}+j} = \frac{\sum_{k=1}^{i} \left(\|X^0_{k\tilde{n}+j} - X^0_{(k-1)\tilde{n}+j}\| \right)^e}{\sum_{k=1}^{\tilde{m}-1} \left(\|X^0_{k\tilde{n}+j} - X^0_{(k-1)\tilde{n}+j}\| \right)^e}, \quad i \in \{1, \dots, \tilde{m}-1\},\ j \in \{1, \dots, \tilde{n}\}. \tag{20}$$

The parameter $e \in [0, 1]$ was introduced in the previous section.

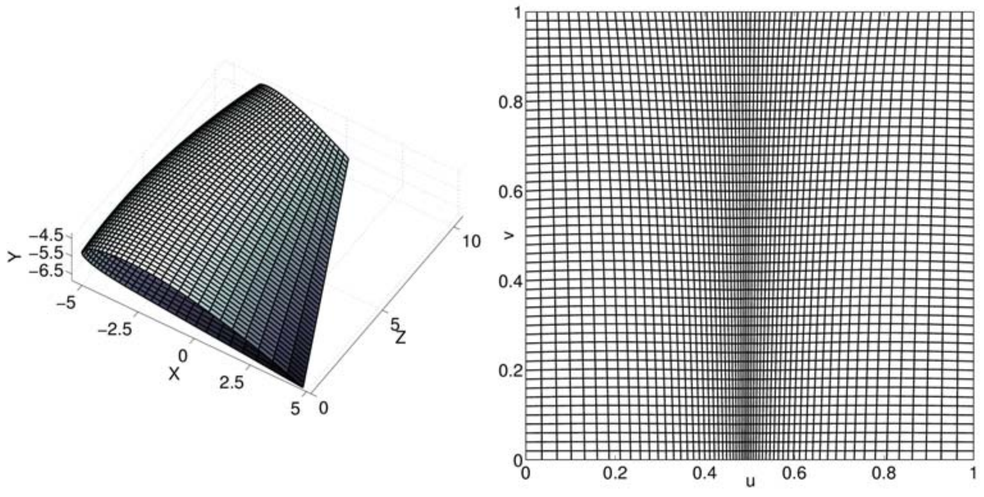

Figure 3. NACA 4415 wing represented via a 3-d structured quadrilateral surface mesh (left) and its parameterization (right)

C. 3-d case with unstructured grids

The parameterization of an unstructured grid which can also be seen as an arbitrary point cloud with four known corner points is very challenging. We like to propose a new and advanced parameterization technique for 3-d wing structures represented with a triangular surface mesh. However, this algorithm is easily adaptable to surface meshes with any kinds of elements.

1. *Definitions*

The following data about the grid must be available; the four corner points

$$C = (C_1, \dots, C_4),\ C_i \in \mathbb{R}^3,\ i \in \{1, \dots, 4\}, \tag{21}$$

the coordinates of each grid node

$$X = (X_1, \dots, X_n),\ X_i \in \mathbb{R}^3,\ i \in \{1, \dots, n\}, \tag{22}$$

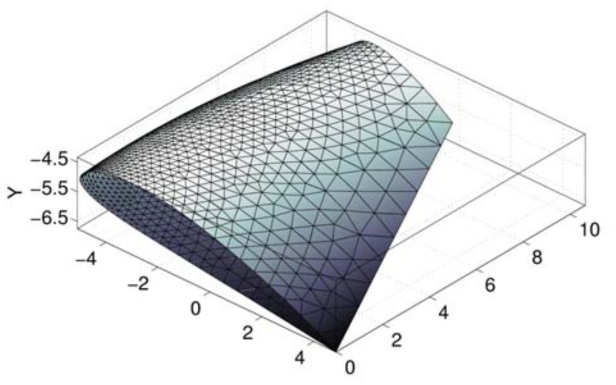

Figure 4. NACA 4415 wing represented via a 3-d unstructured triangular surface mesh

and a list of elements

$$E = (E_1, \dots, E_m),\ E_i = (n_1^i, n_2^i, n_3^i),\ i \in \{1, \dots, m\}, \tag{23}$$

where $n_j^i,\ j \in \{1, \dots, 3\}$ denotes the three node numbers which define the i-th element E_i. Please note that $j = 3$ due to our triangular example, however this whole algorithm is adaptable to any element. The proposed parameterization is possible with just these few information about the grid. Thus, there are no restrictions towards the spatial position, wing shape, angle of attack, sweep angle or twist angle.

2. Derivation of a mean plane

Due to arbitrary twist angles the four corner points are not necessarily onto one plane. Hence, we need to define a mean plane to be able to project all grid point onto a single plane. Thus, we define this plane with two vectors $D_1,\ D_2$ spanned by the diagonal corner points and model point $S^{\square}$, which is defined to be the center of gravity (Eq. (31)) of the rectangle spanned by corner points C. Hence,

$$D = \begin{pmatrix} D_1 \\ D_2 \\ D_3 \end{pmatrix} = \begin{pmatrix} \dfrac{C_3 - C_1}{\|C_3 - C_1\|} \\ \dfrac{C_3 - C_1}{\|C_4 - C_2\|} \\ D_2 \times D_1 \end{pmatrix}, \tag{24}$$

where D_3 is the normal vector of the mean plane. We chose this definition for the mean plane, as it averages the projection error of each point best.

In order to derive $S^{\square}$ we divide the rectangle into two triangles and compute their areas $A_1^{\triangle}, A_2^{\triangle}$ by applying the Heron's formula[15] and their respective center of gravities $S_1^{\triangle}, S_2^{\triangle}$. Let

$$h_1 = \frac{1}{2}\left(\|C_3 - C_1\| + \|C_2 - C_1\| + \|C_4 - C_1\|\right), \tag{25}$$

$$A_1^{\triangle} = \sqrt{h_1(h_1 - \|C_3 - C_1\|)(h_1 - \|C_2 - C_1\|)(h_1 - \|C_4 - C_1\|)} \tag{26}$$

and

$$h_2 = \frac{1}{2}\left(\|C_3 - C_1\| + \|C_4 - C_1\| + \|C_4 - C_3\|\right), \tag{27}$$

$$A_2^{\triangle} = \sqrt{h_2(h_2 - \|C_3 - C_1\|)(h_2 - \|C_4 - C_1\|)(h_2 - \|C_4 - C_3\|)} \tag{28}$$

and

$$S_1^{\triangle} = \frac{1}{3}\left(C_1 + C_2 + C_3\right), \tag{29}$$

$$S_2^{\triangle} = \frac{1}{3}\left(C_1 + C_3 + C_4\right). \tag{30}$$

This leads to the center of gravity of the rectangle

$$S^{\square} = \frac{S_1^{\triangle} A_1^{\triangle} + S_2^{\triangle} A_2^{\triangle}}{A_1^{\triangle} A_2^{\triangle}} \tag{31}$$

which completes the definition of the mean plane (see also Fig. 6).

3. Split wing in lower and upper shell

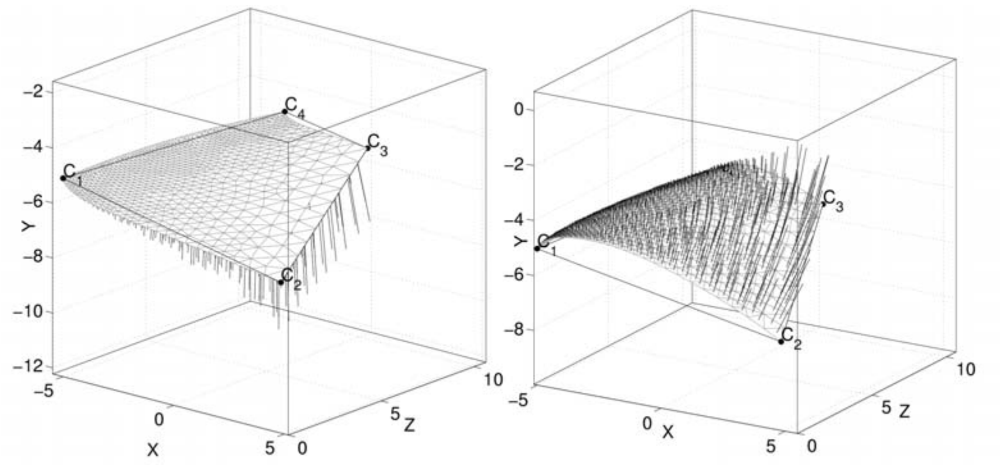

Figure 5. Upper and lower shell of the NACA 4415 wing, its normal vectors $E^{\perp}$ and corner points C

Next we split the wing into an upper and lower shell to make computations easier by treating them separately and patch them together again at the end of the parameterization algorithm. In order to find out which element is part of the upper or lower shell we compute the normal vector

$$E_i^{\perp} = (X^0_{n_2^i} - X^0_{n_1^i}) \times (X^0_{n_3^i} - X^0_{n_1^i}),\ i \in \{1, \dots, m\} \tag{32}$$

of each element as well as its angle towards the normal vector D_3 of the mean plane with

$$\alpha_i = \arccos\left(\frac{E_i^{\perp} D_3^T}{\|E_i^{\perp}\| \|D_3\|}\right),\ i \in \{1, \dots, m\}. \tag{33}$$

Now lets define $\alpha \geq \pi/2$ as upper half and $\alpha < \pi/2$ as lower half. (see Fig. 5)

4. Projection onto mean plane

After defining the mean plane and splitting the wing structure in an upper and lower shell, we project the grid points and corner points onto the mean plane. This is the first step of a series of transformations to map the grid of the wing structure into a $[0, 1] \times [0, 1]$ space. Therefore, the projected corner points are computed by

$$\bar{C}_i = C_i - D_3(A_i D_3^T - S D_3^T),\ i \in \{1, \dots, 4\}. \tag{34}$$

The projected grid points are derived via

$$\bar{X}_i^0 = X_i^0 - D_3(X_i D_3^T - S D_3^T),\ i \in \{1, \dots, n\}. \tag{35}$$

Figure 6 illustrates C and $\bar{C}$, as well as D.

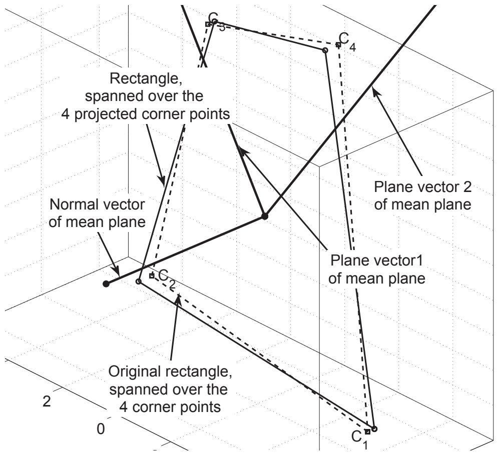
C3
C4
Rectangle, spanned over the 4 projected corner points
Plane vector 2 of mean plane
Normal vector of mean plane
Plane vector1 of mean plane
C2
Original rectangle, spanned over the 4 corner points
2
0
C1

Figure 6. Mean plane of the 3-d NACA 4415 wing, the original corner points, the mean plane vectors and the projected corner points

5. Rotate and translate projection to the origin and the x-z-plane

After projecting all grid points onto the mean plane we need to eliminate one spatial direction of the plane. Thus, we use simple geometric translation and rotation matrices to move all grid points and corner points of the mean plane to the origin and into the x-z-plane. This process is illustrated in Fig. 7. Please note, to simplify matters and to keep illustrations clear, we just show the grid of the upper shell throughout the next sections. Let

$$\hat{X}^0 = (\hat{X}^0_1, \dots, \hat{X}^0_n),\ \hat{X}^0_i = \begin{pmatrix} \hat{x}^0_i \\ \hat{y}^0_i = 0 \\ \hat{z}^0_i \end{pmatrix},\ i \in \{1, \dots, n\} \tag{36}$$

and

$$\hat{C} = (\hat{C}_1, \dots, \hat{C}_4),\ \hat{C}_i = \begin{pmatrix} \hat{x}^c_i \\ \hat{y}^c_i = 0 \\ \hat{z}^c_i \end{pmatrix},\ i \in \{1, \dots, 4\},\ \text{where } \hat{C}_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{37}$$

be the rotated and translated grid and corner points.

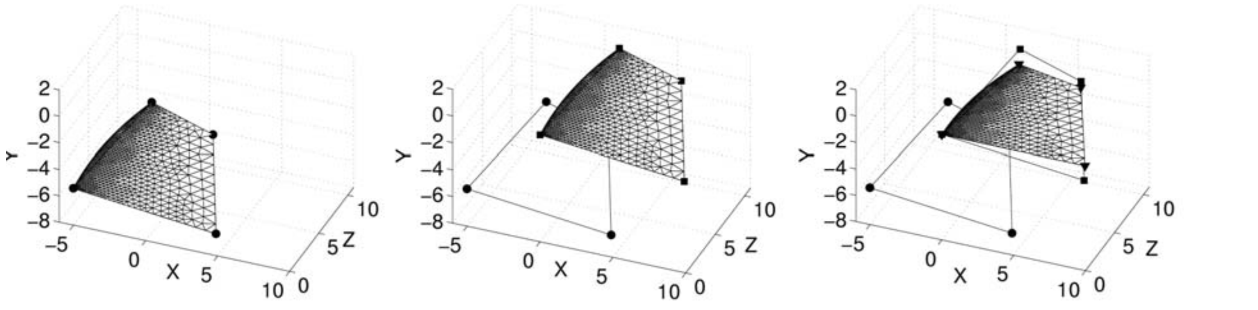

Figure 7. Corner points of the mean plane (left, circles), translation of point $\bar{C}_1$ to origin (middle, squares), rotation into the x-z-plane (right,triangles)

6. Mapping into u-v-space

The translated and rotated corner points $\hat{C}$ are used as boundary conditions for a linear mapping from the x-z-plane into the u-v-space. This is done by solving

$$M \begin{pmatrix} \hat{u}_i \\ \hat{v}_i \end{pmatrix} = \begin{pmatrix} \hat{x}^0_i \\ \hat{z}^0_i \end{pmatrix},\ i \in \{1, \dots, n\}, \tag{38}$$

where

$$M = \begin{pmatrix} \hat{x}_2^c & \hat{x}_4^c \\ \hat{z}_2^c & \hat{z}_4^c \end{pmatrix}. \tag{39}$$

This will map corner point $\hat{C}_2 \rightarrow [1,0]$ and $\hat{C}_4 \rightarrow [0,1]$ into the u-v-space. Hence,

$$\begin{pmatrix} \hat{u}_i \\ \hat{v}_i \end{pmatrix} = M^{-1} \begin{pmatrix} \hat{x}_i^0 \\ \hat{z}_i^0 \end{pmatrix}, \; i \in \{1, \dots, n\}, \tag{40}$$

which can be seen in Fig. 8. Furthermore, we need to map the position of corner point $\hat{C}_3$ into the u-v-space with

$$\begin{pmatrix} C_3^u \\ C_3^v \end{pmatrix} = M^{-1} \begin{pmatrix} \hat{x}_3^c \\ \hat{z}_3^c \end{pmatrix}. \tag{41}$$

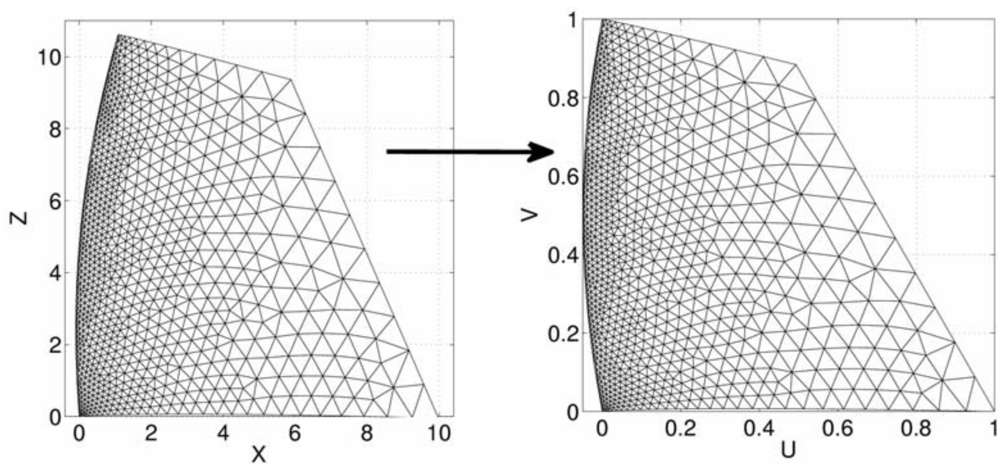

Figure 8. Map from the x-z-space to the u-v-space

7. Modification of Batina's spring analogy

Up to this point we achieved that three out of four corner points are at their correct position in the u-v-space, but our final goal is to fit the boundaries of each wing shell exactly into a $[0,1] \times [0,1]$ space. Thus, we propose a two-step procedure based on a modified spring analogy algorithm which was first developed by Batina,[16].[17] The first step is the movement of corner point $(C_3^u, C_3^v) \rightarrow [1,1]$ using the spring analogy. The second step is the movement of all boundary nodes to their respective boundaries of $[0,1] \times [0,1]$. The following spring analogy algorithm can only be applied to one shell at a time. Therefore, to keep things simple, we define $\hat{u}^*, \hat{v}^*$ to be the parameters $\hat{u}, \hat{v}$ of one shell and n^* the number of nodes on one shell.

Before we present those two steps, we like to introduce our modified spring analogy in general. Let

$$\tilde{N}^{all} = (\tilde{N}_1, \dots, \tilde{N}_{n^*}), \text{ where } \tilde{N}_i \in \mathbb{N}, \; i \in \{1, \dots, n^*\} \tag{42}$$

be a set of all global node numbers corresponding to the coordinates in X. In addition we define

$$\tilde{N}^{in} = (\tilde{N}_1^{in}, \dots, \tilde{N}_f^{in}), \; \tilde{N}_i^{in} \in \{\tilde{N}^{all}\}, \; i \in \{1, \dots, f\} \tag{43}$$

to be a set of f interior node numbers, whit $\{\tilde{N}^{in}\} \subseteq \{N^{all}\}$ and

$$\tilde{N}^{ex} = (\tilde{N}_1^{ex}, \dots, \tilde{N}_{n^*-f}^{ex}), \; \tilde{N}_i^{ex} \in \{\tilde{N}^{all}\}, \; i \in \{1, \dots, n^* - f\} \tag{44}$$

to be set of $n^* - f$ exterior node numbers, with $(\{\tilde{N}^{ex}\} \subseteq \{N^{all}\}) \wedge (\{\tilde{N}^{ex}\} \cap \{\tilde{N}^{in}\} = \{\emptyset\})$.

We distinguish between interior and exterior nodes, as we propose to know the displacement of exterior nodes and we compute, using the spring analogy, the displacements of the interior nodes.

By using the information of our element matrix E as well as $\tilde{N}^{all}, \tilde{N}^{in}, \tilde{N}^{ex}$ we create the following matrices

$$L_i^{in} = (L_{i,1}^{in}, \ldots, L_{i,g_i}^{in}),\ L_{i,j}^{in} \in \{\tilde{N}^{in}\},\ j \in \{1, \ldots, g_i\},\ i \in \{1, \ldots, f\}, \tag{45}$$

which denote a list of g_i neighboring (connected via an edge) interior node numbers of the respective i-th interior node. Furthermore we define

$$L_i^{ex} = (L_{i,1}^{ex}, \ldots, L_{i,h_i}^{ex}),\ L_{i,j}^{ex} \in \{\tilde{N}^{ex}\},\ j \in \{1, \ldots, h_i\},\ i \in \{1, \ldots, f\}, \tag{46}$$

to be a list of h_i neighboring (connected via an edge) exterior node numbers of the respective i-th interior node.

Now we compute the spring stiffness of all edges which have at least one interior node by assuming that the stiffness is inversely proportional to the edge length. Note that it is very important to initialize the edge stiffness matrix with zeros by applying

$$k_{i,j} = 0,\ i, j \in \{1, \ldots, n^*\}. \tag{47}$$

This is physically correct as we assume that the stiffness of any node pair which is not connected via an edge is zero. However, the edge stiffness of existing edges is computed via

$$k_{\tilde{N}_i^{in}, L_{i,j}^{in}} = \frac{1}{\sqrt{\left(\hat{u}^*_{L_{i,j}^{in}} - u^*_{N_i^{in}}\right)^2 + \left(\hat{v}^*_{L_{i,j}^{in}} - \hat{v}^*_{N_i^{in}}\right)^2}},\ i \in \{1, \ldots, f\},\ j \in \{1, \ldots, g_i\} \tag{48}$$

$$k_{\tilde{N}_i^{in}, L_{i,j}^{ex}} = \frac{1}{\sqrt{\left(\hat{u}^*_{L_{i,j}^{ex}} - u^*_{N_i^{in}}\right)^2 + \left(\hat{v}^*_{L_{i,j}^{ex}} - \hat{v}^*_{N_i^{in}}\right)^2}},\ i \in \{1, \ldots, f\},\ j \in \{1, \ldots, h_i\}. \tag{49}$$

Furthermore we define a global displacement vector $\delta = (\delta_i)_{i=1,\ldots,n^*},\ \delta_i \in \mathbb{R}^2$ and store the displacements of the exterior nodes with

$$\delta_{\tilde{N}_i^{ex}} = \delta_i^{ex},\ i \in \{1, \ldots, n^* - f\}. \tag{50}$$

Finally we can compute the interior node displacements with

$$K\delta^{in} = F, \tag{51}$$

where $K = (K_{i,j})_{i,j},\ i, j \in \{1, \ldots, f\}$ denotes the stiffness matrix and $F = (F_i)_{i=1,\ldots,f}$ the force vector. The stiffness matrix is defined as

$$K_{i,j|_{i=j}} = \sum_{l=1}^{f} k_{\tilde{N}_i^{in}, l} \text{ and} \tag{52}$$

$$K_{i,j|_{i\neq j}} = -k_{\tilde{N}_i^{in}, \tilde{N}_j^{in}}. \tag{53}$$

The force vector is computed with

$$F_i = \sum_{j=1}^{h_i} k_{\tilde{N}_i^{in}, L_{i,j}^{ex}} \delta_{L_{i,j}^{ex}}. \tag{54}$$

To compute the interior displacements we solve Eq. (51) for δ^{in} and apply

$$\delta_{\tilde{N}_i^{in}} = \delta_i^{in},\ i \in \{1, \ldots, f\}, \tag{55}$$

which yields to a complete displacement vector δ. Hence, we compute our new grid point locations in the u-v-space by

$$\begin{pmatrix} u_i^* \\ v_i^* \end{pmatrix} = \delta_i + \begin{pmatrix} \hat{u}_i^* \\ \hat{v}_i^* \end{pmatrix},\ i \in \{1, \ldots, n^*\}. \tag{56}$$

After explaining the general procedure we will now present the actual displacement of corner point $(C_3^u, C_3^v) \rightarrow [1, 1]$ as well as the movement of the wing's surface boundaries onto $[0, 1] \times [0, 1]$.

8. Move corner point $(C_3^u, C_3^v) \rightarrow [1, 1]$

To move the third corner point we apply the algorithm of the previous section and define that the corner points are exterior nodes and all other nodes are interior ones:

$$\tilde{N}^{ex} = (\tilde{N}^{ex}_{C_1}, \tilde{N}^{ex}_{C_2}, \tilde{N}^{ex}_{C_3}, \tilde{N}^{ex}_{C_4}), \tag{57}$$

where $\tilde{N}^{ex}_{C_i}$ denotes the node number of the i-th corner point. Furthermore we define

$$\begin{pmatrix} \delta_{\tilde{N}^{ex}_{C_1}} \\ \delta_{\tilde{N}^{ex}_{C_2}} \\ \delta_{\tilde{N}^{ex}_{C_3}} \\ \delta_{\tilde{N}^{ex}_{C_4}} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ (1 - C_3^u, 1 - C_3^v) \\ 0 \end{pmatrix} \tag{58}$$

to be the external nodes displacements. Applying Eq. (45)-(56) to the upper and lower shell yield to correct positions of all four corner points (see Fig. 9).

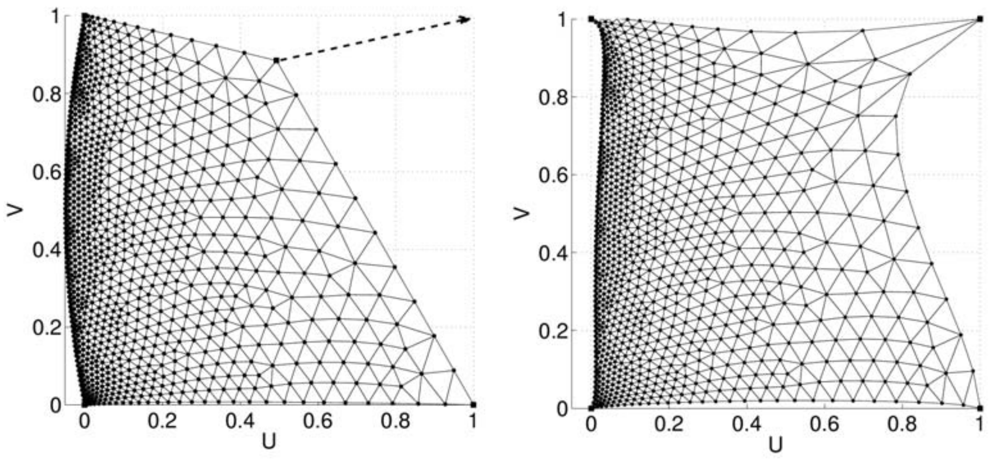

Figure 9. Applying the spring analogy with corner points as exterior nodes (left: squares are exterior points, circles are interior points; right: after application of the algorithm)

9. Fit boundaries to $[0, 1] \times [0, 1]$ *space*

After all corner points reached their final positions only the fitting of the grid's boundaries to $[0, 1] \times [0, 1]$ is left. Therefore, we need to find all exterior nodes. This can be done, e.g. by creating an edge list with every edge of every element by using E. Interior edges will appear twice, whereas exterior edges only once. This gives us the information whether a node is exterior or interior. However, we also need to know the node's boundary number as there are four boundaries and it is essential to know on which boundary we like to displace our exterior node. There are a couple of ways to find out on which boundary the node is. One way can be, e.g. to compute the normal vector of each exterior edge and the angles towards the u- and v-axis. With certain ranges and combinations of angles we find the corresponding boundary. Thus, we can compute all δ^{ex} (see left Fig. 10).

Finally we can apply Eq. (45)-(56) to the upper and lower shell and compute the interior displacements (see right Fig. 10).

10. Combine lower and upper shell

The last step is the combination of the upper and lower shell. Therefore, let $(\hat{u}_i^{up}, \hat{v}_i^{up})_{i \in \{1,\ldots,n^{up}\}}$ correspond the upper shell and let $(\hat{u}_i^{low}, \hat{v}_i^{low})_{i \in \{1,\ldots,n^{low}\}}$ correspond to the lower shell. Hence, to combine those into a

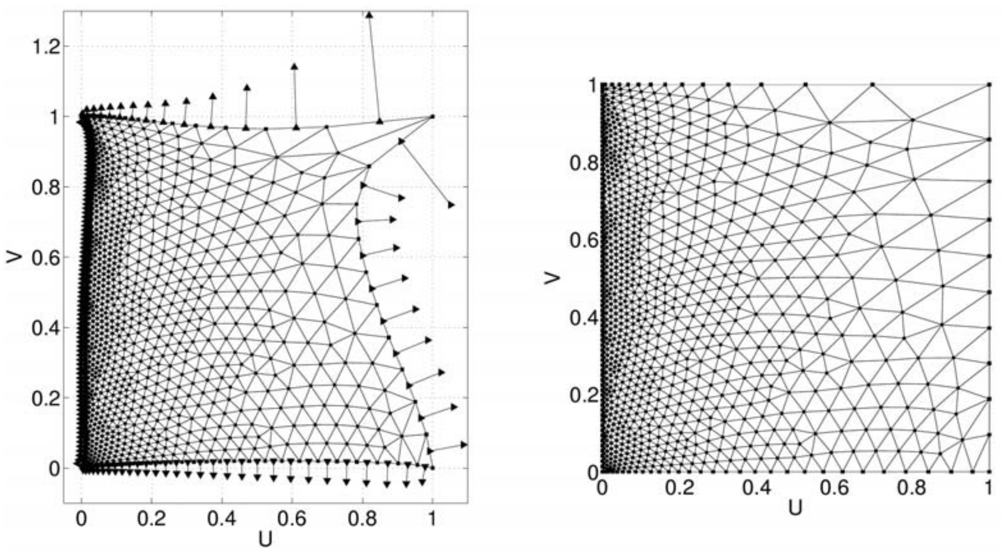

Figure 10. Applying the spring analogy with boundary nodes as exterior nodes (left: squares are exterior points, circles are interior points as well as the vectors pointing towards the corresponding boundary; right: after application of the algorithm)

$[0,1] \times [0,1]$ space we apply

$$u_i = \frac{1}{2}(1 - \hat{u}_i^{low}),\ i \in \{1, \ldots, n^{low}\}, \tag{59}$$

$$v_i = \frac{1}{2}(1 - \hat{v}_i^{low}),\ i \in \{1, \ldots, n^{low}\} \tag{60}$$

and

$$u_{i+n^{low}} = \frac{1}{2}\hat{u}_i^{up} + 0.5,\ i \in \{1, \ldots, n^{up}\}, \tag{61}$$

$$v_{i+n^{low}} = \frac{1}{2}\hat{v}_i^{up} + 0.5,\ i \in \{1, \ldots, n^{up}\}. \tag{62}$$

Hence, (u, v) are our final parameterization, illustrated in Fig. 11.

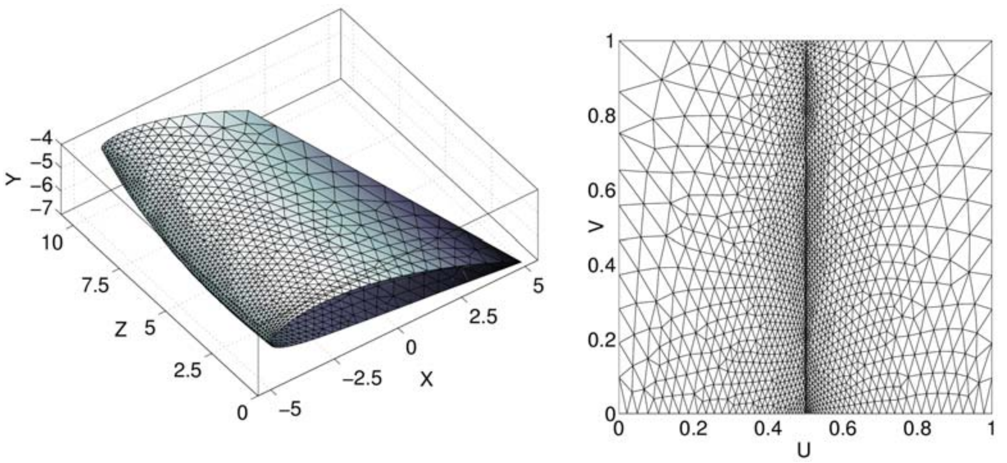

Figure 11. Left: NACA 4415 wing represented via a 3-d unstructured triangular grid, right: final parameterization

V. Knot vector computation

In this section we present an automatic knot vector computation based on the parameterization of the given grid points. We distinguish between 2-d and 3-d grids in this section. In general a knot vector defines the relative positions along the parameterization at which the representations of each curve segment are patched together. Furthermore, the knot vector allows, by repeating entries, a local reduction of the basis function's order. Thus, we are able to coincide the start and end control point onto the start and end curve point by repeating the first and last entry of the knot vector p or q times respectively.

A. 2-d case

We use parameters u to compute our knot vector $\xi = (\xi_j)_{j=1,\dots,\hat{n}+p}$, as we assume that the given curve point distribution represents also the complexity of the curve, ergo where we need a higher density of control points. Hence, the knot vector is defined as

$$\xi = (\underbrace{0,\dots,0}_{p}, \xi_{p+1},\dots,\xi_{\hat{n}}, \underbrace{1,\dots,1}_{p}) \tag{63}$$

where the remaining entries are computed using following algorithm which is based on Ref. 6

$$\begin{aligned} I &= \frac{n-1}{\hat{n}-p+1} \\ j &= \lfloor Ii+1 \rfloor \\ R &= Ii+1-j \\ \xi_{p+i} &= (1-R)u_j + Ru_{j+1} \end{aligned} \tag{64}$$

with $i = 1,\dots,\hat{n}-p$. Figure 12 illustrates the parameterization of a 2-d NACA 4415 wing (see Fig. 2) with $e = 1$ and its corresponding knot vector $\xi = (0,0,0,0,0.25,0.39,0.47,0.52,0.6,0.74,1,1,1,1)$ for basis functions of order $p = 4$ and $\hat{n} = 10$ control points.

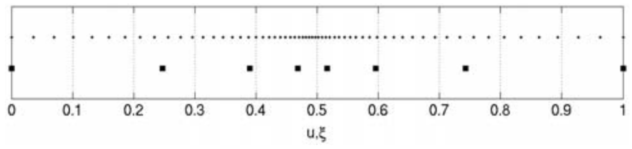

Figure 12. Parameterization of the 2-d NACA 4415 wing (see Fig. 2) and the corresponding knot vector ξ

B. 3-d case

In the 3-d case we do not distinguish between the grid types and propose one simple scheme for all. We use our parameterization u, v again as basis to compute our knot vector $\xi = (\xi_j)_j$, $j = 1,\dots,\hat{n}+p$ in the u-direction and $\nu = (\nu_j)_j$, $j = 1,\dots,\hat{m}+q$ in the v-direction. We propose an extension of the algorithm of Eq. (64), but at first we sort u and v ascending into $\tilde{u}, \tilde{v}$ and define

$$\xi = (\underbrace{0,\dots,0}_{p}, \xi_{p+1},\dots,\xi_{\hat{n}}, \underbrace{1,\dots,1}_{p}) \tag{65}$$

$$\nu = (\underbrace{0,\dots,0}_{q}, \nu_{q+1},\dots,\nu_{\hat{m}}, \underbrace{1,\dots,1}_{q}) \tag{66}$$

Now by applying Eq. (64)

$$\begin{aligned} I &= \frac{n-1}{\hat{n}-p+1} \\ j &= \lfloor Ii+1 \rfloor \\ R &= Ii+1-j \\ \xi_{p+i} &= (1-R)\tilde{u}_j + R\tilde{u}_{j+1} \end{aligned} \tag{67}$$

with $i = 1, \ldots, \hat{n} - p$ as well as

$$\begin{aligned} I &= \frac{n-1}{\hat{m} - q + 1} \\ j &= \lfloor Ii + 1 \rfloor \\ R &= Ii + 1 - j \\ \nu_{q+i} &= (1 - R)\tilde{v}_j + R\tilde{v}_{j+1} \end{aligned} \tag{68}$$

with $i = 1, \ldots, \hat{m} - q$. Figure 13 illustrates on the left side the parameterization of the structured, quadrilateral grid of the 3-d NACA 4415 wing (see Fig. 3) with $e = 1$ and its corresponding knot vectors $\xi = (0, 0, 0, 0, 0.22, 0.37, 0.46, 0.53, 0.62, 0.77, 1, 1, 1, 1)$ for basis functions of order $p = 4$ and $\hat{n} = 10$ control points in the u-direction as well as knot vector $\nu = (0, 0, 0, 0.2, 0.4, 0.6, 0.8, 1, 1, 1)$ for basis functions of order $q = 3$ and $\hat{m} = 7$ control points in the v-direction. On the right side the parameterization of the same wing based on an unstructured, triangular grid (see Fig. 11) is shown with the corresponding knot vectors $\xi = (0, 0, 0, 0, 0.39, 0.47, 0.5, 0.51, 0.53, 0.61, 1, 1, 1, 1)$ for basis functions of order $p = 4$ and $\hat{n} = 10$ control points in the u-direction and knot vector $\nu = (0, 0, 0, 0.14, 0.33, 0.52, 0.75, 1, 1, 1)$ for basis functions of order $q = 3$ and $\hat{m} = 7$ control points in the v-direction.

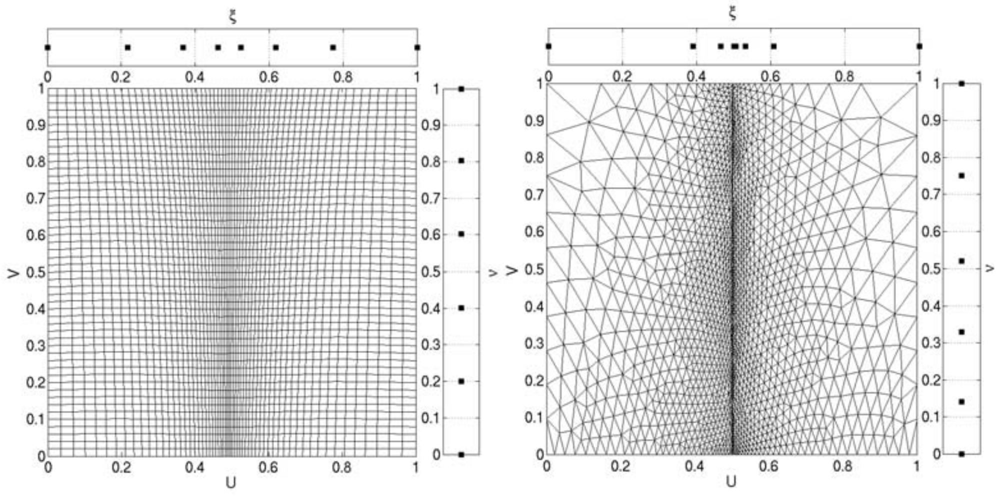

Figure 13. Left: Parameterization of the 3-d NACA 4415 wing (see Fig. 3) and its corresponding knot vectors ξ, ν; right: Parameterization of the 3-d NACA 4415 wing (see Fig. 11) and its corresponding knot vectors ξ, ν

VI. Reduction to a linear regression problem and its solution

After parameterizing the given grid points and computing the corresponding knot vectors, this section presents the reduction to a linear regression problem and its solution towards the unknown control point positions. The weights and the exact location of the control points can be solved later in a second step as an optimization problem. This will be presented in the next section.

A. Reduction to a linear problem

The first step towards the solution of our linear regression problem is the application of the Cox-deBoor recurrence[12] (see Eq. (2)) to compute all necessary basis functions.

Lets rewrite Eq. (9) and Eq. (17) with $w_i = 1, \ \forall i$ as

$$X_i^0 = \sum_{j=1}^{\hat{n}} P_j^0 N_{i,j}^p, \quad i \in \{1, \ldots, n\}, \tag{69}$$

for the 2-d case and

$$X_i^0 = \sum_{j=1}^{\hat{k}} P_j^0 N_{i,j}, \quad i \in \{1, \dots, n\}. \tag{70}$$

for the 3-d case, where $\hat{k} = \hat{n}\hat{m}$.

To simplify matters we rewrite these two equations into the matrix form

$$X^0 = N^p P^0 \tag{71}$$

for the 2-d case, where X^0 denotes the $n \times 2$ matrix of the given set of points, N^p denotes the basis functions ($n \times \hat{n}$ matrix) and P^0 the control points ($\hat{n} \times 2$ matrix). The 3-d case is rewritten as

$$X^0 = NP^0, \tag{72}$$

where X^0 denotes the $n \times 3$ matrix of the given set of points, N denotes the basis functions ($n \times \hat{k}$ matrix) and P^0 the control points ($\hat{k} \times 3$ matrix).

B. Solving the overdetermined system

We need to solve those systems for the unknown P^0. There are many ways to solve linear regression problems, we have chosen a Single Value Decomposition (SVD). We present the SVD only for the 2-d case. It is straightforwardly adaptable to 3-d. However, we need to solve this overdetermined system by computing the inverse of the non-quadratic matrix N^p by decomposing it into the following matrices by using its eigenvalues and -vectors:

$$\begin{aligned} U &= \mathrm{eig}(N^p (N^p)^T), && n \times n \\ D &= \sqrt{\mathrm{diag}(\mathrm{eig}(N^p (N^p)^T))} && n \times \hat{n} \\ V^T &= \mathrm{eig}((N^p)^T N^p), && \hat{n} \times \hat{n} \end{aligned}$$

such that

$$N^p = UDV^T. \tag{73}$$

In order to compute $(N^p)^{-1}$ we apply

$$(N^p)^{-1} = V^T D^{-1} U^T \tag{74}$$

where

$$(D^{-1})^T = \frac{1}{D_{i,i}}, \quad i = 1, \dots, \hat{n}.$$

Finally this leads to the inital control point positions

$$P_j^0 = (N^p)^{-1}_{j,i} X_i^0, \ \forall i, \quad j = 1, \dots, \hat{n}. \tag{75}$$

The same applies for the 3-d case by replacing $\hat{n}$ with $\hat{k}$ and N^p with N.

C. Examples

Finally we like to present some examples in 2-d and 3-d, with structured and unstructured grids. Figure 14 shows the approximated NACA 4415 wing in 2-d, where the control points are computed with the proposed algorithm, using the given grid points and parameterization shown in Fig. 2 and the subsequent knot vector ξ. We used an order of $p = 3$ and set the number of control points to $\hat{n} = 9$. Compared to the given grid the average deviation of the approximation is $4.6e - 3$ per node.

Figure 15 illustrates the approximated 3-d NACA 4415 wing based on the quadrilateral, structured grid (on the left) and its parameterization shown in Fig. 3. We set the number of control points to $\hat{n} = 11 \times \hat{m} = 5$ and applied the proposed algorithm with orders $p = 4, q = 3$ as well as the computed knot vectors ξ, ν. The

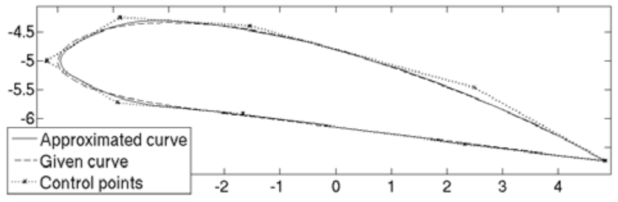

Figure 14. NACA 4415 wing in 2-d, approximated with the proposed algorithm.

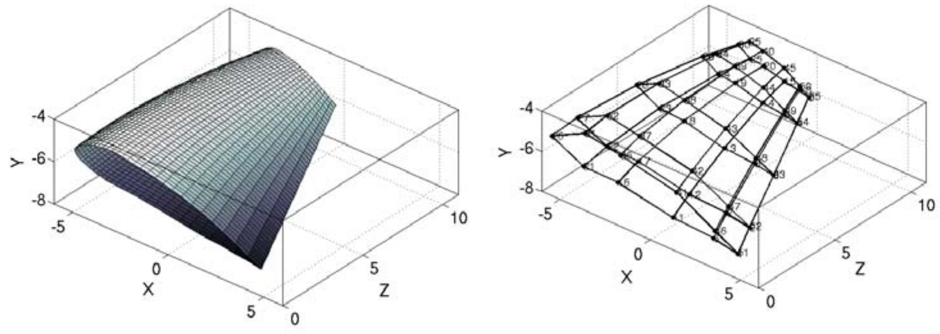

Figure 15. NACA 4415 wing in 3-d (left), approximated as NURBS surface based on the structured quadrilateral parameterization with the computed control points (right)

resulting control point positions are shown on left. The average deviation of the approximation towards the given grid is $5.1e-04$ per node.

Figure 16 presents the NURBS surface approximation of the NACA 4415 wing based on the unstructured, triangular grid and its parameterization as shown in Fig. 11 (left). The average deviation of this approximation to the given grid is $6.1e-04$ per node. We set the number of control points to $\hat{n} = 11 \times \hat{m} = 5$ and computed the control points with orders $p = 4, q = 3$ and the subsequent knot vectors ξ, ν. The resulting control point positions of the linear regression problem are shown in the middle. In the right part a uniformly spaced parameterization with 100×50 points and its resulting knot vectors is applied. Note that we used the same control point positions as well as orders of the B-Spline functions. This would be the used representation in a CAD software.

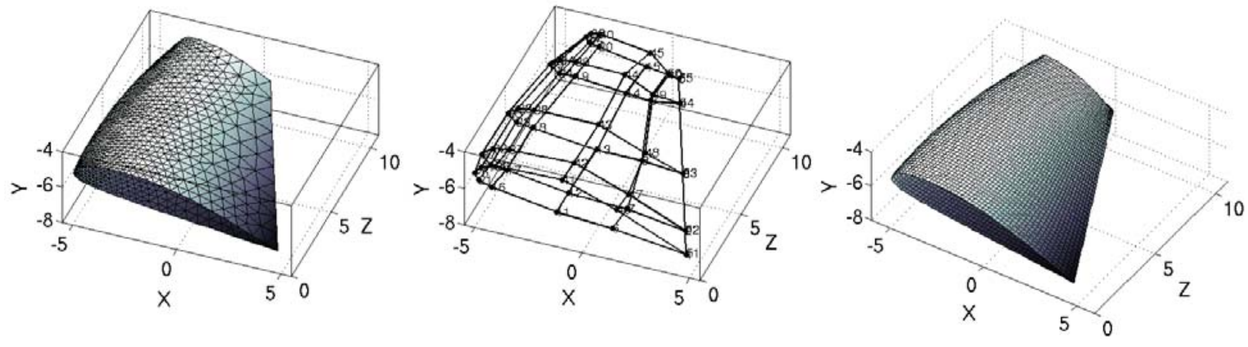

Figure 16. NACA 4415 wing in 3-d, approximated as NURBS surface based on the unstructured triangular parameterization (left) with the computed control points (middle) and an approximated NURBS surface based on an uniform parameterization.

VII. Optimizing weights and location of control points

To be able to approximate, e.g., circular shapes we need to improve the weight as well as to optimize the position of the initial control points P^0 gained by the algorithm already presented within this paper. However, the optimization is optional as the results of the linear regression problem are usually satisfying. Nevertheless, for the sake of completeness we propose the following simple optimization algorithm. We define a matrix of design parameters for the 3-d case as follows

$$a = (a_{k,\Theta})_{k,\Theta}, \; k = \hat{N}^l_\Theta, \dots, \hat{N}^u_\Theta \quad \Theta = 1, \dots, 4 \tag{76}$$

Please note that the 2-d case needs just a simple modification of the presented 3-d algorithm by neglecting the third dimension. $\Theta = 1$ denotes the design parameters altering the weights and $\Theta = 2$, $\Theta = 3$ and $\Theta = 4$ denotes the relation to either the x, y or z-component of the control points.

A. Objective function

We choose to minimize the sum of the quadratic distances between the given set of points $X^0_i = (x^0, y^0, z^0)_i, \; i \in \{1, \dots, n\}$ and the NURBS surface approximation $X_i = (x, y, z)_i, \; i \in \{1, \dots, n\}$. This leads to the following objective function

$$\begin{aligned} \min_a J &= \gamma \sum_{i=1}^{n} \left(\left[x_i(a) - x^0_i\right]^2 + \left[y_i(a) - y^0_i\right]^2 + \left[z_i(a) - z^0_i\right]^2 \right) \\ &\text{s.t. } lb_{k,\Theta} \le a_{k,\Theta} \le ub_{k,\Theta}, \forall k \quad \Theta = 1, \dots, 4 \end{aligned} \tag{77}$$

with scaling factor γ.

B. Gradient of the objective function

If we want to be able to use a gradient-based optimization approach we need to compute the derivatives of (77):

$$\frac{\partial J}{\partial a_{k,\Theta}} = \alpha \sum_{i=1}^{n} \left(2\left[x_i - x^0_i\right] \frac{\partial x_i}{\partial a_{k,\Theta}} + 2\left[y_i - y^0_i\right] \frac{\partial y_i}{\partial a_{k,\Theta}} + 2\left[z_i - z^0_i\right] \frac{\partial z_i}{\partial a_{k,\Theta}} \right) \tag{78}$$

where $k = \hat{N}^l_\Theta, \dots, \hat{N}^u_\Theta$, $\Theta = 1, \dots, 4$ and

$$\frac{\partial x_i}{\partial a_{k,\Theta}} = \begin{cases} \dfrac{P^x_k N_{i,k} \sum_{g=1}^{\hat{n}} w_g N_{i,g} - \left(\sum_{g=1}^{\hat{n}} P^x_g w_g N_{i,g}\right) N_{i,k}}{\left(\sum_{g=1}^{\hat{n}} w_g N_{i,g}\right)^2} & \text{, if } \Theta = 1 \\ \dfrac{w_k N_{i,k}}{\sum_{g=1}^{\hat{n}} w_g N_{i,g}} & \text{, if } \Theta = 2 \\ 0 & \text{, if } \Theta = 3 \\ 0 & \text{, if } \Theta = 4 \end{cases}, \tag{79}$$

$$\frac{\partial y_i}{\partial a_{k,\Theta}} = \begin{cases} \dfrac{P^y_k N_{i,k} \sum_{g=1}^{\hat{n}} w_g N_{i,g} - \left(\sum_{g=1}^{\hat{n}} P^y_g w_g N_{i,g}\right) N_{i,k}}{\left(\sum_{g=1}^{\hat{n}} w_g N_{i,g}\right)^2} & \text{, if } \Theta = 1 \\ 0 & \text{, if } \Theta = 2 \\ \dfrac{w_k N_{i,k}}{\sum_{g=1}^{\hat{n}} w_g N_{i,g}} & \text{, if } \Theta = 3 \\ 0 & \text{, if } \Theta = 4 \end{cases}, \tag{80}$$

$$\frac{\partial z_i}{\partial a_{k,\Theta}} = \begin{cases} \dfrac{P_k^z N_{i,k} \sum_{g=1}^{\hat{n}} w_g N_{i,g} - \left(\sum_{g=1}^{\hat{n}} P_g^z w_g N_{i,g}\right) N_{i,k}}{\left(\sum_{g=1}^{\hat{n}} w_g N_{i,g}\right)^2} & \text{, if } \Theta = 1 \\ 0 & \text{, if } \Theta = 2 \\ 0 & \text{, if } \Theta = 3 \\ \dfrac{w_k N_{i,k}}{\sum_{g=1}^{\hat{n}} w_g N_{i,g}} & \text{, if } \Theta = 4 \end{cases} . \tag{81}$$

The objective function and the gradient can be applied to any suitable optimization procedure to optimize the control point positions as well as the weights towards the best possible fit.

C. Example

Fig. 17 shows a 2-d example of an gradient-based optimization, where an improvement of 62% was reached due to 66 optimization loops applying a Trust-Region derivative-based optimizer. The initial set of control points of the NACA 4415 wing 2-d grid is computed with the proposed linear regression algorithm and the parameterization shown in Fig. 2 with its resulting knot vector ξ. For the optimization procedure all control points and all weights are chosen to be optimized. The average deviation of the initial approximation towards the given grid is $4.6e-3$ per node, after the optimization the average deviation decreased to $1.7e-3$. Furthermore the new weights are shown in the figure. It easy to see that the improvement of the weights are important for representing circular shapes with NURBS.

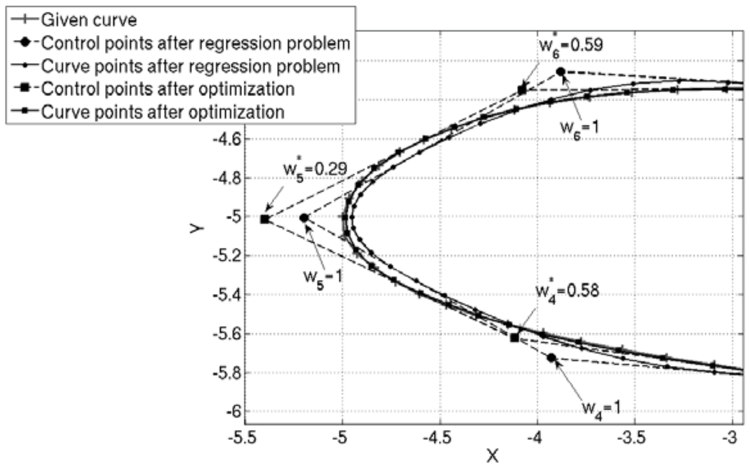

Figure 17. Approximated and optimized NACA 4415 wing in 2-d with the proposed procedure.

VIII. Conclusion

We proposed a new and advanced tool for fitting NURBS curves and surfaces to sets of given grid points throughout this paper. The computation of the control point coordinates for surface grids in 2-d as well as structured and unstructured grids in 3-d is presented via a two-step algorithm. The first step includes the parameterization of the given grid points as well as the automatic knot vector and basis function computations. Then the NURBS curve or surface definition is reduced to a linear regression problem, which can be solved for the unknown control point positions. A good parameterization and suitable knot vectors are crucial for a successful completion of the first step. In a second step an optimization problem was defined including the gradient computation of the objective function to optimize the control point positions and to set up the optimal weights. This allows us to even represent circular shapes, which can not be represented exactly by the reduced NURBS formulation. Furthermore, the functionality of the presented algorithms is shown by using the example of a twisted NACA 4415 wing with different grid representations in 2-d and 3-d.

References

[1]Jameson, A., “Efficient Aerodynamic Shape Optimization,” *Proceedings of the 10th AIAA/ISSMO Multidisciplany Analysis and Optimization Conferencein Albany, New York, August 30 - September 1, 2004*, AIAA Paper 2004-4369, 2004.

[2]Becker, G., Falk, U., and Schäfer, M., “Shape Optimization with Higher-Order Surfaces in Consideration of Fluid-Structure Interaction,” *Fluid-Structure Interaction: Theory, Numerics and Applications*, Kassel University Press, Kassel, 2009.

[3]Becker, G., Siegmann, J., Michaelis, J., and Schäfer, M., “Comparison of a derivative-free and a gradient-based shape optimization method in the context of fluid-structure interactions,” *8th World Congress on Structural and Multidisciplinary Optimization*, 2009.

[4]Ma, W. and Kruth, J.-P., “NURBS curve and surface fitting for reverse engineering,” *The International Journal of Advanced Manufacturing Technology*, Vol. 14, No. 12, 1998, pp. 918–927.

[5]Jung, H. and Kim, K., “A new parameterisation method for NURBS surface interpolation,” *The International Journal of Advanced Manufacturing Technology*, Vol. 16, No. 11, 2000, pp. 784–790.

[6]Dill, L. H., “Representation of Ice Geometry by Parametric Functions: Construction of Approximating NURBS Curves and Quantification of Ice Roughness-Year 1: Approximating NURBS Curves,” Tech. Rep. NASA/CR-2004-213071, University of Akron, April 2004.

[7]Piegl, L., “On NURBS: A survey,” *Computer Graphics and Applications, IEEE*, Vol. 11, No. 1, Jan 1991, pp. 55–71.

[8]Piegl, L. and Tiller, W., “Curve and surface constructions using rational B-splines,” *Computer-Aided Design*, Vol. 19, No. 9, Nov. 1987, pp. 485–498.

[9]Rogers, D. F. and Adams, J. A., *Mathematical elements for computer graphics*, McGraw-Hill, New York, 1976.

[10]Tiller, W., “Rational B-Splines for Curve and Surface Representation,” *Computer Graphics and Applications, IEEE*, Vol. 3, No. 6, Sept. 1983, pp. 61–69.

[11]Zeid, I., *Mastering CAD-CAM*, McGraw–Hill Higher Education, Boston, 2005.

[12]De Boor, C., *A practical guide to splines*, Springer, 2001.

[13]Farin, G. E., Hoschek, J., and Kim, M. S., editors, *Handbook of computer aided geometric design*, Elsevier, Amsterdam, 2002.

[14]Piegl, L. and Tiller, W., *The NURBS book*, Springer Verlag, New York, 1996.

[15]Eves, H. and Eves, J., *An introduction to the history of mathematics*, Holt, Rinehart and Winston, 1969.

[16]Batina, J., “Unsteady Euler airfoil solutions using unstructured dynamic meshes,” *AIAA Journal*, Vol. 28, No. 8, 1990, pp. 1381–1388.

[17]Blom, F., “Considerations on the spring analogy,” *International Journal for Numerical Methods in Fluids*, Vol. 32, No. 6, 2000, pp. 647–668.