# Volume Mesh Generation for Numerical Flow Simulations using Catmull-Clark and Surface Approximation Methods

Michael Rom[1,2] and Karl-Heinz Brakhage[1]

[1] Institut für Geometrie und Praktische Mathematik, RWTH Aachen, 52056 Aachen, Germany, {rom,brakhage}@igpm.rwth-aachen.de
[2] German Research School for Simulation Sciences, 52425 Jülich, Germany

**Summary.** In this paper we present a new technique for the semi-automatic generation of volume meshes which can be used for numerical flow simulations. Our aim is to end up with a high-quality block-structured volume mesh connected to a smooth surface mesh. For this purpose we start with a polyhedron giving a rough approximation of the target surface geometry which can be of arbitrary genus. To this initial polyhedron we apply an alternating iterative process of Catmull-Clark subdivisions and approximations of the target surface. Usually, such a surface is given by a collection of trimmed B-spline surfaces. Hence, the best approximation results can be achieved by projecting the points of the Catmull-Clark limit surface, which converges to uniform bi-cubic B-spline patches, onto it and subsequently recomputing the control mesh. If we construct another polyhedron surrounding the initial polyhedron, both being automatically connected to each other, we can perform three-dimensional Catmull-Clark subdivision to the flow field between the inner and the outer surface.

## 1 Introduction, Significance and Related Work

Numerical flow simulations require high-quality volume meshes. The generation of such meshes as well as the construction of the object around or through which the flow should be simulated can be very difficult and time-consuming. We have developed a new promising approach by combining established methods for 2D- and 3D-subdivision with new or adapted algorithms for the computation of the limit points of a subdivision surface, the projection of these limit points onto given B-spline surfaces and the approximation of these given surfaces to get new surface mesh control points (= vertices).

The overall process is illustrated in Fig. 1. The user only has to construct a simple inner initial surface polyhedron, being roughly similar to the target surface of arbitrary genus, and an outer surrounding polyhedron by giving the coordinates, the face connectivity and optionally edges which should be

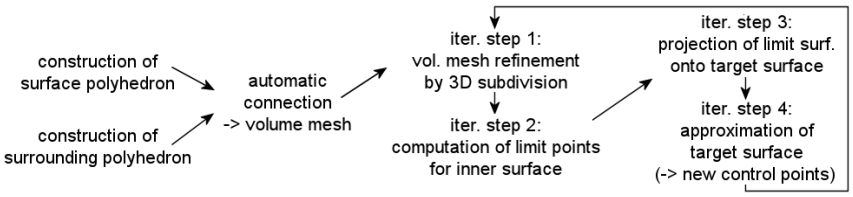


**Fig. 1.** Process of volume mesh generation

treated like creases. These two surfaces are connected automatically leading to a coarse volume mesh representing the flow field. This initial semi-automatic step is illustrated for a simple wing with an open end for the attachment of a fuselage in Fig. 2.

The subsequent iterative process can be stopped if the approximation of the target surface is satisfying. If necessary, a more uniform distribution of mesh control points can be obtained by applying smoothing steps after an approximation step in each iteration. The mesh resulting from the overall process depicted in Fig. 1 can be converted to a B-spline volume mesh. Hence, it is possible to apply further grid refinement by spline evaluation.

Figure 3 gives a 1D-example of the iterative process by showing a fragment of a polygon: $\mathbf{P}_0$ is a vertex of this control polygon. One subdivision (iteration step 1) leads to a new position of that vertex ($\mathbf{Q}_0$) and a refinement of the control polygon by inserting edge points (which are not drawn in the picture). The point on the limit curve corresponding to $\mathbf{P}_0$ can be calculated and is denoted by $\mathbf{L}_0$ (iteration step 2). If this is projected to a given target curve we obtain $\mathbf{L}_p$ (iteration step 3). To force our limit curve to pass through $\mathbf{L}_p$ or at least to lie close to it, we have to replace $\mathbf{Q}_0$ by $\mathbf{Q}_1$. This calculation is done by a least squares approximation (iteration step 4). Repeating the overall process results in an accurate approximation of the target curve along with a consequent refinement of control points.

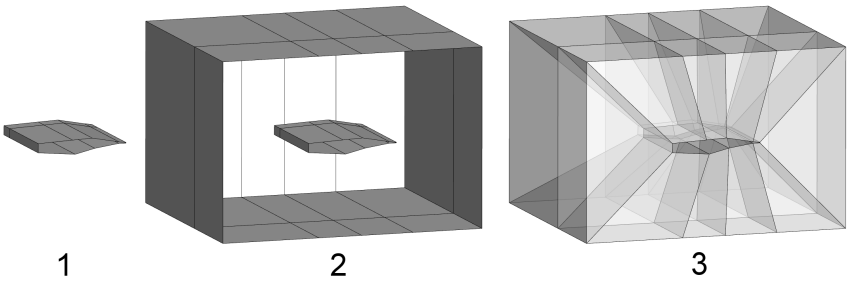


**Fig. 2.** Semi-automatic construction of an initial mesh (1: simple polyhedron as a starting point for the volume mesh generation, 2: surrounding polyhedron around the initial one, 3: edge and face connections between the inner and the outer polyhedron, translucent view)

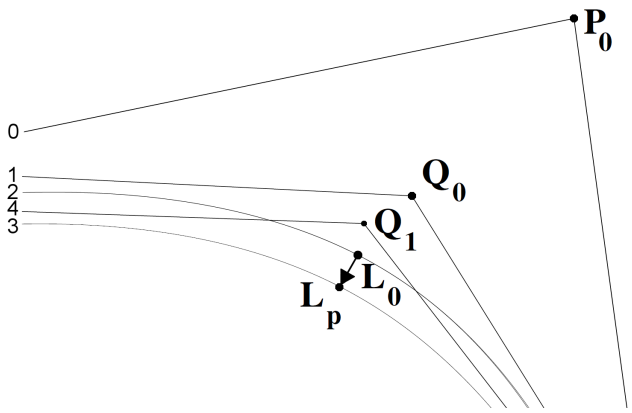


**Fig. 3.** Example for the workflow in the case of a curve (0: initial control polygon, 1: control polygon after one subdivision, 2: limit curve of the subdivision process, 3: given target curve, 4: new control polygon after approximation of the target curve)

For the 2D-subdivisions of the inner and the outer surface, which are applied simultaneously to the 3D-subdivision of the space in between, we use the scheme presented by Catmull and Clark [1] which is applicable to surfaces of arbitrary genus. Due to the convergence of the Catmull-Clark limit surface to uniform bi-cubic B-spline patches (see [1]) we get a smooth $C^2$-surface with the exception of points where no tensor-product topology is given ($C^1$ there). The results of Stam's analysis of the subdivision matrix [2] with an extension for modeling creases by de Rose et al. [3] allow for the pre-calculation of the limit points of the inner surface at each refinement level. The limit points can then be used for the approximation of a given target surface. In our practical applications these target surfaces are often given as trimmed B-spline surfaces. The Nelder-Mead optimization algorithm [4] is used for the projection of the limit surface onto such a B-spline surface.

The space between the inner and the outer surface is the flow field and has to be refined successively. Thus, we need an extension of the Catmull-Clark subdivision rules to volume meshes. The first scheme for that purpose was presented by Joy and MacCracken [5] as a direct generalization of the two-dimensional subdivision rules. The analysis of the rules in matters of the smoothness of the resulting volume mesh is very difficult. Hence, Bajaj et al. [6] tried another approach: They factored the tri-cubic subdivision process into tri-linear subdivision followed by averaging, overall leading to rules which are easier to understand. We apply 3D-subdivision by using a combination of these two schemes.

The rest of the paper is structured as follows: Section 2 gives an overview of the Catmull-Clark subdivision rules for surfaces and volumes and describes why the Catmull-Clark scheme is the method of choice for our purpose. In

Sect. 3 we briefly explain how the points of the limit surface can be calculated and demonstrate how they are used for the approximation of target surfaces given by a B-spline representation. Section 4 describes a non-shrinking mesh smoothing algorithm which can be applied after an approximation to obtain a more uniform distribution of mesh control points.

## 2 Background on Catmull-Clark Subdivision

In iteration step 1 of our mesh generation process (see Fig. 1) two-dimensional Catmull-Clark subdivision [1] is applied to the inner and to the outer surface, while the space in between is refined by using derived rules for three-dimensional subdivision.

### 2D-Subdivision

Catmull and Clark published their descriptions of quadratic and cubic subdivision surfaces in 1978. Contrary to tensor-product splines, this scheme can be applied to meshes that are not regular rectangular grids. A refinement step is defined by the following rules:

1. For each face add a point given by the average of the $N$ face vertices:

$$\mathbf{F} = \frac{1}{N} \sum_{i=0}^{N-1} \mathbf{P}_i$$

2. For each edge add a point given by a weighted average of the two new adjacent face points $\mathbf{F}_{left}$, $\mathbf{F}_{right}$ and the edge midpoint $\mathbf{E}_c$:

$$\mathbf{E} = \frac{1}{4}(\mathbf{F}_{left} + 2\mathbf{E}_c + \mathbf{F}_{right})$$

3. Move each old vertex to a new position given by a weighted average of the vertex $\mathbf{P}$, the average $\tilde{\mathbf{F}}$ of the new adjacent face points and the average $\tilde{\mathbf{E}}_c$ of the adjacent edge midpoints:

$$\mathbf{P}_{new} = \frac{1}{N}(\tilde{\mathbf{F}} + 2\tilde{\mathbf{E}}_c + (N-3)\mathbf{P})$$

   $N$ denotes the valence of $\mathbf{P}$, i.e., the number of edges connected to $\mathbf{P}$.

Figure 4 shows three fragments of a mesh representing these three refinement rules of one Catmull-Clark subdivision. The weights given for the vertices and face points directly result from the rules. Finally, the new edges are built by splitting the old ones and connecting the new face points to the new adjacent edge points. The new faces are the faces inside the old ones.

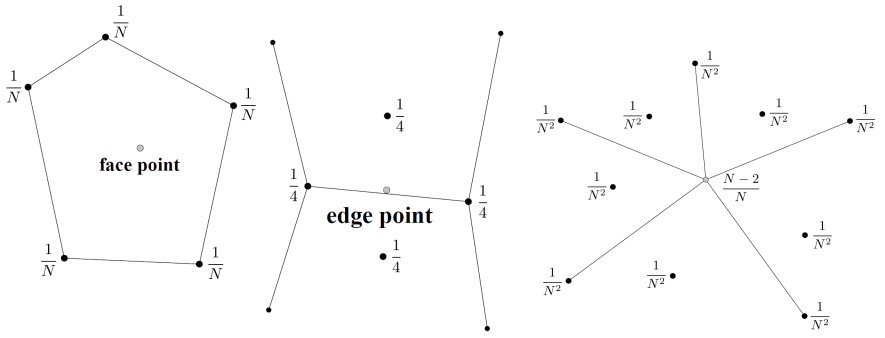


**Fig. 4.** Illustration of the subdivision rules for the Catmull-Clark method, from left to right: insertion of a new face point, insertion of a new edge point, computation of a new vertex position

Applying the subdivision rules to a mesh of quadrilaterals we obtain a refined mesh at the new subdivision level $l+1$ with the following numbers of faces ($\#F$), edges ($\#E$) and vertices ($\#V$):

$$\begin{aligned} \#F_{l+1} &= 4\ \#F_l \\ \#E_{l+1} &= 4\ \#F_l + 2\ \#E_l \\ \#V_{l+1} &= \#F_l + \#E_l + \#V_l \end{aligned}$$

In our modeling and grid generation concepts (see [7]) we want to end up with smooth untrimmed B-spline patches. These can be provided by an easy implementable conversion of the Catmull-Clark limit surface. Thus, this scheme is the method of choice. We summarize its crucial properties (cf. [8]):

- The surfaces can be of arbitrary genus since the subdivision rules can be carried out to a mesh of arbitrary topological type.
- After one subdivision all faces are quadrilaterals.
- Except at extraordinary vertices (vertices of valence $N \neq 4$) the limit surface converges to uniform bi-cubic B-spline patches. Hence, the surface is $C^2$-continuous except at extraordinary vertices.
- Near an extraordinary vertex the surface can be shown to have a well defined tangent plane at the limit point, but the curvature there is generally not well defined.
- The number of extraordinary vertices is fixed after the first subdivision, so that less smooth regions are scaled down with each further subdivision (see Sect. 3 and Fig. 7 for more details).
- After two subdivisions each face can contain one extraordinary vertex at most. This allows us to easily compute the points of the Catmull-Clark limit surface, see Sect. 3.
- The subdivision rules can be modified in such a way that they generate infinitely sharp creases as well as semi-sharp creases, i.e., creases for which

the sharpness can vary from zero (meaning smooth) to infinite, see de Rose et al. [3] for more details. An example for a situation in which we need the possibility of modeling creases is the connection of a wing to a fuselage which should not be smoothed during a subdivision process.

### Example

Figure 5 gives an example for the application of the Catmull-Clark subdivision scheme. Despite using a rough approximation of a wing as an initial mesh, we end up with a smooth mesh of high quality after only a few subdivisions (in this case three).

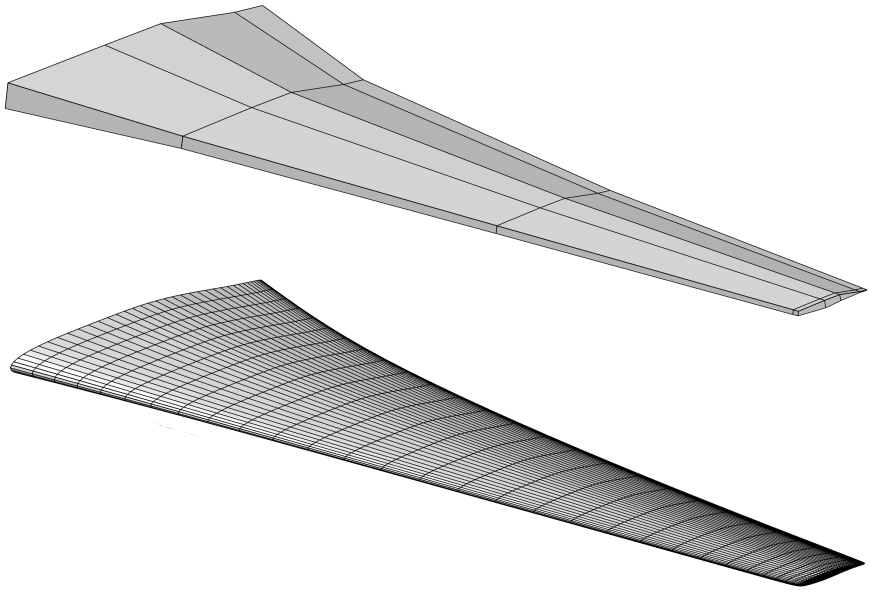

**Fig. 5.** Initial polyhedron for the Catmull-Clark method (top) and polyhedron after three subdivisions (bottom)

### 3D-Subdivision

For three-dimensional subdivision we use the rules given in the list below demanding the volume mesh to contain only hexahedra composed of eight vertices, six faces and twelve edges. This is guaranteed if the inner surface mesh exclusively consists of quadrilaterals before connecting it to the outer one. Since the first Catmull-Clark subdivision always provides a quadrilateral mesh, this requirement can be fulfilled easily.

The following rules are used:

1. For each volumetric cell add a point given by the average of the eight cell vertices:
$$\mathbf{C} = \frac{1}{8}\sum_{i=0}^{7}\mathbf{P}_i$$
2. For each face add a point given by a weighted average of the two new adjacent cell points $\mathbf{C}_{left}$, $\mathbf{C}_{right}$ and the face centroid $\mathbf{F}_c$:
$$\mathbf{F} = \frac{1}{4}(\mathbf{C}_{left} + 2\mathbf{F}_c + \mathbf{C}_{right})$$
3. For each edge add a point given by a weighted average of the edge midpoint $\mathbf{E}_c$ where $N$ denotes the number of adjacent faces, the average $\tilde{\mathbf{C}}$ of the new adjacent cell points and the average $\tilde{\mathbf{F}}_c$ of the adjacent face centroids:
$$\mathbf{E} = \frac{1}{N}(\tilde{\mathbf{C}} + 2\tilde{\mathbf{F}}_c + (N-3)\mathbf{E}_c)$$
4. Move each old vertex to a new position:
$$\mathbf{P}_{new} = \sum_{\mathbf{U}_i \in ring(\mathbf{P})} \frac{3^{3-dim(\mathbf{U}_i,\mathbf{P})}}{4^3\ val(\mathbf{P})}\, val(\mathbf{U}_i,\mathbf{P})\,\mathbf{U}_i$$
with $ring(\mathbf{P})$ denoting the set of vertices $\mathbf{U}_i$ connected to $\mathbf{P}$ by a cell, a face or an edge. The vertex $\mathbf{P}$ itself is stored in $ring(\mathbf{P})$, too. $val(\mathbf{P})$ is the number of cells containing $\mathbf{P}$, whereas $val(\mathbf{U}_i, \mathbf{P})$ gives the number of cells containing both $\mathbf{U}_i$ and $\mathbf{P}$. $dim(\mathbf{U}_i, \mathbf{P})$ specifies the connection between $\mathbf{U}_i$ and $\mathbf{P}$ (0 if $\mathbf{U}_i = \mathbf{P}$, 1 if $\mathbf{U}_i$ and $\mathbf{P}$ lie on a common edge, 2 if they only lie on a common face or 3 if they only lie on a common cell).

The first three of these 3D-rules were presented by Joy and MacCracken [5] in 1996. Since their vertex recomputation formula (rule 4) does not consider an adjustment for the number of edges a vertex is connected to, we use the equation published by Bajaj et al. [6] in 2002 instead.

Finally, the new edges are built by splitting the old ones, connecting the new face points to the new adjacent edge points and connecting the new cell points to the new adjacent face points. The new faces are the faces inside the old ones and inside the old cells. The new cells are the cells inside the old ones. Similar to the surface case we can define special edges (creases) and special faces to which the subdivision coefficients for curves and for surfaces are applied, respectively.

Since our volume meshes consist of hexahedra only (see above), each subdivision leads to the following numbers of cells ($\#C$), faces ($\#F$), edges ($\#E$) and vertices ($\#V$) at the new subdivision level $l+1$:

$$
\begin{aligned}
\#C_{l+1} &= 8\ \#C_l \\
\#F_{l+1} &= 12\ \#C_l + 4\ \#F_l \\
\#E_{l+1} &= 6\ \#C_l + 4\ \#F_l + 2\ \#E_l \\
\#V_{l+1} &= \#C_l + \#F_l + \#E_l + \#V_l
\end{aligned}
$$

**Example**

An example for three-dimensional subdivision is given in Fig. 6. The coarse volume mesh in the right of Fig. 2 (mesh 3) has been subdivided twice treating the outer surface vertices as fixed and the outer surface edges as creases.

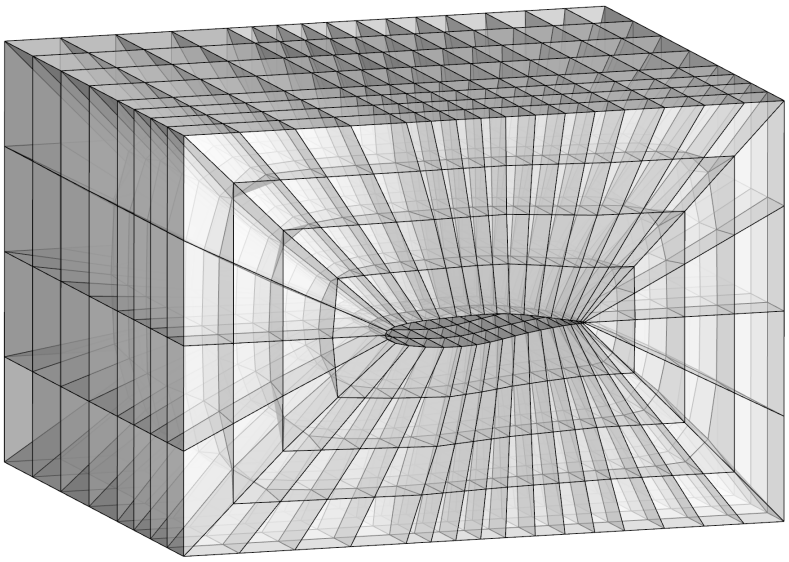

**Fig. 6.** Example for three-dimensional subdivision (translucent view)

## 3 Limit Points of Catmull-Clark Subdivision Surfaces, Projection onto B-Spline Surfaces and Approximation

Stam [2] gave an algorithm for evaluating the Catmull-Clark scheme and its derivatives at arbitrary points. He used a choice of ordering for the control vertices such that the main part of the subdivision matrix has a cyclical structure. Hence, the discrete Fourier transform can be used to compute its eigenstructure. For the modified rules given in [3] this analysis is very technical, details of the investigation and the implementation can be found in [9].

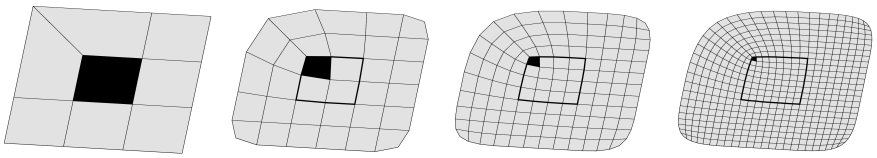

**Fig. 7.** Behavior near an extraordinary vertex of valence $N = 3$

Due to the existence of extraordinary points a surface mesh cannot be evaluated everywhere at each subdivision level by well-known B-spline algorithms because the control vertex structure near an extraordinary point is not a simple rectangular grid. Thus, all faces that contain extraordinary vertices cannot be evaluated as uniform B-splines. For our mesh we assume that each face is a quadrilateral and contains one extraordinary vertex at most, which both is fulfilled after two subdivisions at the latest. Figure 7 shows that the region in which the surface cannot be evaluated with standard methods is scaled down with every subdivision.

Since we can evaluate the surface away from extraordinary vertices as a regular bi-cubic B-spline, the remaining problem is to demonstrate how to evaluate a patch corresponding to a face with just one extraordinary vertex, such as the dark region shown in Fig. 7. We introduce parameter values and define a surface patch $\mathbf{x}(u, v)$ over the unit square $[0, 1] \times [0, 1]$ such that the point $\mathbf{x}(0, 0)$ corresponds to the extraordinary vertex. We can evaluate the surface at such a vertex ($\mathbf{x}(0, 0)$) as a linear combination of the circumfluent vertices. Additionally, we can evaluate $\mathbf{x}(u, 1)$ for $u \in [0, 1]$ and $\mathbf{x}(1, v)$ for $v \in [0, 1]$ as regular B-spline part. The remaining problem is the evaluation $\mathbf{x}(u, v)$ in the rest of the unit square. This problem is solved by doing just enough subdivisions such that $(u, v)$ corresponds to a regular part at that stage to do the evaluation as a regular bi-cubic B-spline.

### Limit Points

A point $\mathbf{L}_i$ of the Catmull-Clark limit surface corresponding to a mesh control point $\mathbf{P}_i^{(k)}$ of the current subdivision level $k$ is defined by

$$\mathbf{L}_i = \lim_{k \to \infty} \mathbf{P}_i^{(k)},$$

i.e., the position after $k \to \infty$ subdivisions. For the approximation of given surfaces with our Catmull-Clark meshes we can pre-compute the values $\mathbf{L}_i$ of the limit surface (see iteration step 2 in Fig. 1) by using Stam's above-mentioned algorithm for evaluating the Catmull-Clark scheme at arbitrary points. For each face we compute nine limit points: four corresponding to the face vertices, four corresponding to the edges and one corresponding to the

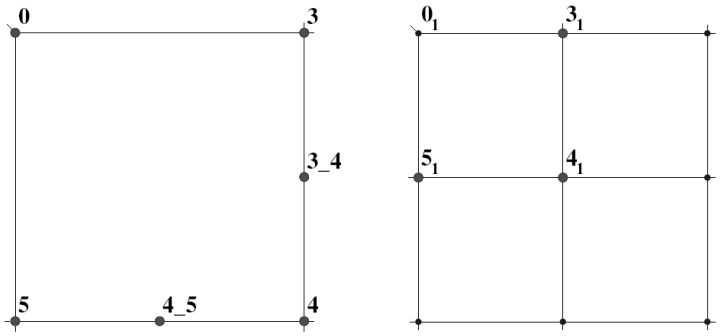


**Fig. 8.** Example for the computation of the nine limit points for a face with one extraordinary vertex (number 0), left: six limit points can be calculated immediately, right: the remaining three limit points can be calculated after one subdivision

face. An example is given in Fig. 8 where the vertex with the number 0 represents an extraordinary vertex. The six limit points for the vertices marked in the left part of the picture can be calculated immediately, since they belong to $\mathbf{x}(0,0)$, $\mathbf{x}(u,1)$ for $u \in [0,1]$ or $\mathbf{x}(1,v)$ for $v \in [0,1]$, see above. The computation of the remaining three limit points requires one subdivision of the mesh, so that the new vertices with the numbers $3_1$, $4_1$ and $5_1$ belong to $\mathbf{x}(u,1)$ for $u \in [0,1]$ or $\mathbf{x}(1,v)$ for $v \in [0,1]$ at the new refinement level. All these computations can be applied without explicitly subdividing and lead to coefficients for each vertex participating in the limit point computation. Hence, the limit points $\mathbf{L}_i$ can be written as a linear combination

$$\mathbf{L}_i = \mathbf{c}_i^T \, \mathbf{V}^i$$

where we have collected the weights $c_j$ of the involved vertices $\mathbf{P}_j^{(k)}$ in the vicinity of the control point $\mathbf{P}_i^{(k)}$ in the vector $\mathbf{c}_i$ and the associated vertices $\mathbf{P}_j^{(k)}$ in $\mathbf{V}^i$. Details of the computation of the coefficients $c_j$ and the choice of the associated vertices $\mathbf{P}_j^{(k)}$ can be found in [9]. The vertex and edge limit points are only calculated if they have not already been computed by considering an adjacent face before. Overall this leads to $\#V_l + \#F_l + \#E_l$ limit points at the current subdivision level $l$. If a control point belongs to a special edge, i.e. a crease, we need other rules for the evaluation. Therefore, we use our extension [9] in these cases.

### Projection

For a given surface $s$ we can project $\mathbf{L}_i$ onto it: $\mathbf{L}_i \rightarrow \mathbf{L}_i^s$ (iteration step 3 in Fig. 1). Usually, the surface $s$ is given by a B-spline representation defined by

$$\mathbf{x}(u,v) = \sum_{r=0}^{m} \sum_{s=0}^{n} \mathbf{x}_{r,s} N_{r,p}(u) N_{s,q}(v) \qquad (\star)$$

where $(m+1)\times(n+1)$ gives the number of control points and $p$ and $q$ denote the degree of the B-spline basis functions $N_{r,p}(u)$ and $N_{s,q}(v)$, respectively. Together with knot vectors in $u$- and $v$-direction, which have been omitted in $(\star)$, we can evaluate $\mathbf{x}(u,v)$ and the partial derivatives $\mathbf{x}_u(u,v)$ and $\mathbf{x}_v(u,v)$ by applying de Boor's algorithm [10]. Given such a representation, we search for parameter values $u_{L_i}$ and $v_{L_i}$ for each limit point $\mathbf{L}_i$ such that

$$\|\mathbf{x}(u_{L_i}, v_{L_i}) - \mathbf{L}_i\|_2 = \min_{u,v} \|\mathbf{x}(u,v) - \mathbf{L}_i\|_2.$$

We use the Nelder-Mead algorithm [4] to solve this distance minimization problem. This optimization algorithm is simple to understand and to implement, does not need any derivatives and is very robust. For each limit point a simplex, which in this two-dimensional case is a triangle in $(u,v)$-space, is built around the initial values $(u_0, v_0)$ which have to be set once prior to the first projection. Due to the robustness of the algorithm an inaccurate choice of initial values does not destroy the convergence of the algorithm. For each further projection we can calculate better initial values being closer to the resulting values: The limit points corresponding to vertices get the current $(u,v)$-values of the vertex as initial values, whereas the $(u,v)$-values for limit points corresponding to face or edge points are initialized by an average of the $(u,v)$-values of the particular adjacent vertices. In the first step of the Nelder-Mead algorithm the distances $d_j = \|\mathbf{L}_i - \mathbf{x}(u_j, v_j)\|_2,\ j = 0,1,2$, of the limit point $\mathbf{L}_i$ to the B-spline surface points $\mathbf{x}(u_j, v_j)$ calculated for the triangle points $\mathbf{y}_j = (u_j, v_j)^T$ are computed determining the best $(b)$, the second-best $(sb)$ and the worst $(w)$ point of the triangle:

$$d_b = \min_j d_j, \qquad d_{sb} = \min_{j \neq b} d_j, \qquad d_w = \max_j d_j, \qquad j = 0,1,2$$

The corresponding triangle points in are denoted by $\mathbf{y}_b = (u_b, v_b)^T$, $\mathbf{y}_{sb} = (u_{sb}, v_{sb})^T$ and $\mathbf{y}_w = (u_w, v_w)^T$, respectively. After that the midpoint

$$\mathbf{y}_m = \frac{1}{2} \sum_{j \neq w} \mathbf{y}_j, \qquad j = 0,1,2$$

of the triangle edge opposite the worst point is computed. This leads to the reflection point

$$\mathbf{y}_r = 2\,\mathbf{y}_m - \mathbf{y}_w.$$

Depending on the distance $d_r$ of $\mathbf{x}(u_r, v_r)$ to the limit point the triangle is changed by applying the following rules:

- If $d_r < d_{sb}$, replace $\mathbf{y}_w$ by $\mathbf{y}_r$ and restart the algorithm at the first step.
- If $d_{sb} \leq d_r < d_w$, compute the contraction point

$$\mathbf{y}_c = \frac{1}{2}(\mathbf{y}_m + \mathbf{y}_r).$$

  - If $d_c < d_r$, replace $\mathbf{y}_w$ by $\mathbf{y}_c$ and restart the algorithm at the first step.
  - If $d_c \geq d_r$, perform a shrinking transformation (see below).
- If $d_r \geq d_w$, compute the contraction point

$$\mathbf{y}_c = \frac{1}{2}(\mathbf{y}_m + \mathbf{y}_w).$$

  - If $d_c < d_w$, replace $\mathbf{y}_w$ by $\mathbf{y}_c$ and restart the algorithm at the first step.
  - If $d_c \geq d_w$, perform a shrinking transformation (see below).
- Shrinking transformation: Compute two new points

$$\mathbf{y}_1 = \frac{1}{2}(\mathbf{y}_b + \mathbf{y}_{sb}), \quad \mathbf{y}_2 = \frac{1}{2}(\mathbf{y}_b + \mathbf{y}_w)$$

  and replace $\mathbf{y}_{sb}$ by $\mathbf{y}_1$ and $\mathbf{y}_w$ by $\mathbf{y}_2$. Restart the algorithm at the first step.

The algorithm is terminated if the distance between the limit point and the B-spline surface point corresponding to the best triangle point is smaller than a threshold $\varepsilon_1$ or if the distance between the best and the worst triangle point is smaller than a threshold $\varepsilon_2$, i.e., the area of the triangle is very small.

### Approximation

Finally, we get an equation of the form

$$\mathbf{c}_i^T \mathbf{V}^i \stackrel{!}{=} \mathbf{L}_i^s, \qquad \mathbf{L}_i^s = \mathbf{x}(u_{L_i}, v_{L_i}).$$

We use the notation $C\mathbf{V} \stackrel{!}{=} \mathbf{L}^s$ for the approximation problem

$$\|C\mathbf{V} - \mathbf{L}^s\|_2 \rightarrow \min$$

and apply it for a single equation of this system, too. As described above, we have $\#V_l + \#F_l + \#E_l$ limit points at the current subdivision level $l$ and, consequentially, the same number of projection points. Hence, we end up with an over-determined sparse linear system for approximation (iteration step 4 in Fig. 1, $\mathbf{Q}_0 \rightarrow \mathbf{Q}_1$ in Fig. 3) which we solve by applying the **c**onjugate **g**radient method for linear **l**east **s**quares (CGLS or also called CGNR [11]), see below. Solving the approximation problem leads to new control point positions with better approximation properties for the next iteration loop.

### CGLS method

Let $A \in \mathbb{R}^{m \times n}$ with $m \geq n$. It is well known that the solution for the linear least squares problem $\|A\mathbf{x} - \mathbf{b}\|_2 \rightarrow \min$ can be determined by solving the normal equations $A^T A\mathbf{x} = A^T\mathbf{b}$. Furthermore, if $\operatorname{rank}(A) = n$ then $A^T A$ is symmetrical positive definite (spd). Unfortunately, in most cases $A^T A$ is badly

conditioned and for a sparse $A$ usually $A^TA$ is not sparse. The conjugate gradient method (CG) can be reformulated to resolve both problems. We use two residual vectors $\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)}$ and $\mathbf{s}^{(k)} = A^T\mathbf{b} - A^TA\mathbf{x}^{(k)} = A^T\mathbf{r}^{(k)}$. With this notation we can formulate the CGLS-algorithm as follows:

`For` $A \in \mathbb{R}^{m\times n}$ `with` $\operatorname{rank}(A) = n$ `and arbitrary initial vector` $\mathbf{x}^{(0)}$`:`

$\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
$\mathbf{s}^{(0)} = A^T\mathbf{r}^{(0)}$
$\mathbf{d}^{(0)} = \mathbf{s}^{(0)}$
`for` $k = 0, 1, 2, ...$
$\alpha_k = \|\mathbf{s}^{(k)}\|_2^2 \,/\, \|A\mathbf{d}^{(k)}\|_2^2$ `// store` $A\mathbf{d}^{(k)}$
$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k\mathbf{d}^{(k)}$
$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k A\mathbf{d}^{(k)}$
$\mathbf{s}^{(k+1)} = A^T\mathbf{r}^{(k+1)}$
$\beta_k = \|\mathbf{s}^{(k+1)}\|_2^2 \,/\, \|\mathbf{s}^{(k)}\|_2^2$
$\mathbf{d}^{(k+1)} = \mathbf{s}^{(k+1)} + \beta_k\mathbf{d}^{(k)}$
`until stop`

We stop if a maximum number of steps is exceeded (emergency exit) or the residual becomes smaller than a given tolerance $\varepsilon$, i.e., $\|\mathbf{s}^{(k)}\|_2 \leq \varepsilon$. Notice that the cost for one CGLS step is linear in the number of control points.

### Example

The wing depicted in Fig. 9 is used for a test of our iterative process from Fig. 1. The target geometry which we want to approximate consists of two different B-spline representations: The wing with degree 3 in $u$- and degree 1 in $v$-direction is continuously differentiable connected to a wing tip (see Fig. 10) with degree 3 in $u$- and degree 2 in $v$-direction. Figure 11 shows the initial polyhedron which has been constructed manually. The result of the alternating application of 2D-subdivision, limit point computation, projection and approximation can be seen in Fig. 12 for the whole wing after three process iterations, while Fig. 13 reveals a detailed view of the wing tip. The agreement

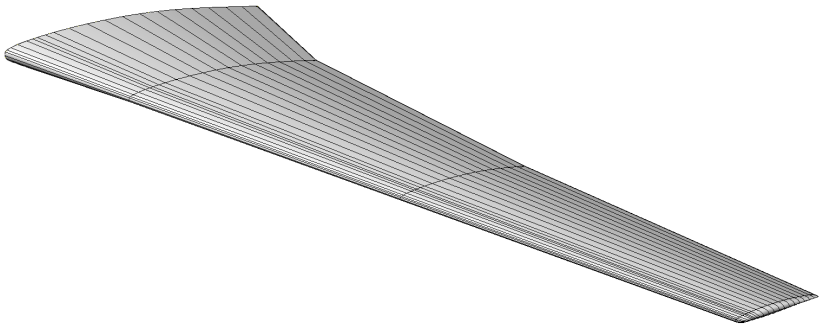

**Fig. 9.** B-spline surface (wing + wing tip) with knot-isolines

of the given B-spline surface and the approximated surface mesh as well as the distribution of mesh points are fine. An even better distribution can be achieved by applying smoothing steps after an approximation as shown in Sect. 4.

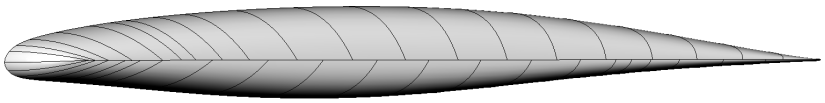

**Fig. 10.** B-spline surface (wing tip) with knot-isolines

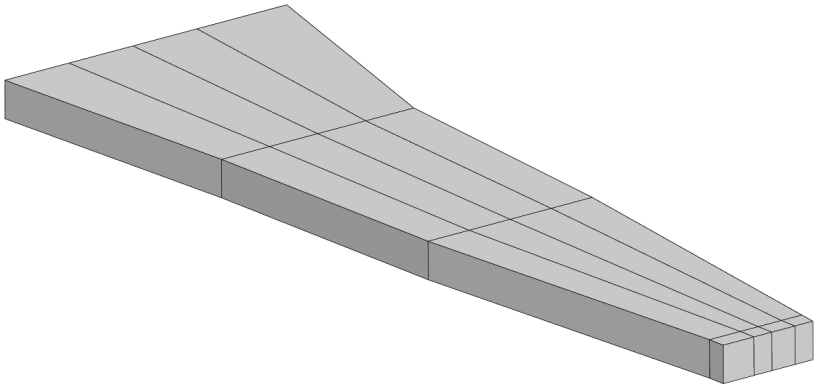

**Fig. 11.** Initial polyhedron for the approximation of the B-spline surface from Fig. 9

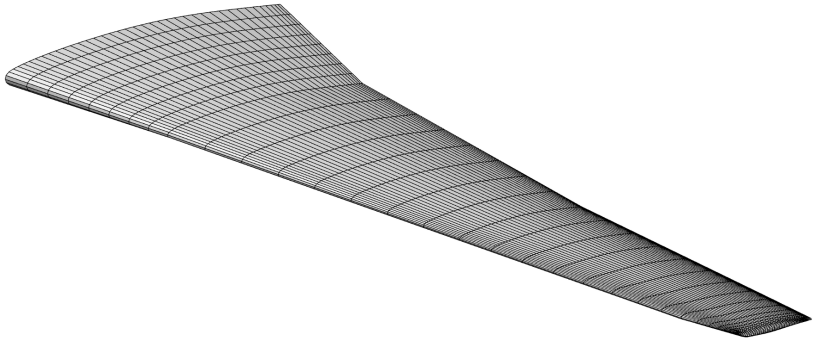

**Fig. 12.** Surface mesh (wing + wing tip) after three process iterations

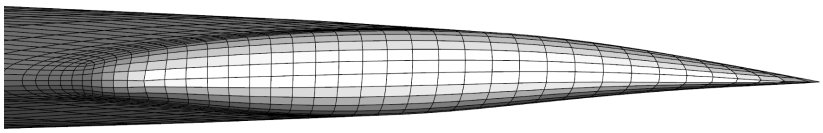

**Fig. 13.** Surface mesh (close-up of wing tip) after three process iterations

## 4 Smoothing

We want to obtain a more uniform distribution of mesh control points. Therefore, we use a smoothing method which is non-shrinking because otherwise we would lose the approximation of the given surface geometry. For each regular control point $\mathbf{P}_i$ of the mesh, i.e., each control point with valence $N = 4$, we search for two opposite adjacent control points $\mathbf{P}_{adj,1}$ and $\mathbf{P}_{adj,3}$ connected by edges and construct a parabola passing through these two opposite vertices and the control point itself. The first smoothing step is then applied by moving the control point $\mathbf{P}_i$ along the parabola into the direction of the midpoint of the parabola. The second step uses the other direction, i.e., passing a parabola through the control point $\mathbf{P}_i$ and its remaining two opposite neighbors $\mathbf{P}_{adj,2}$ and $\mathbf{P}_{adj,4}$ and again moving into the direction of the midpoint of that new parabola. An example for the procedure is depicted in Fig. 14.

The derivation of the computations needed for a smoothing step can be explained on the basis of Fig. 15. The equation for the planar parabola passing through the three points $\mathbf{P}_{adj,1}$, $\mathbf{P}_i$ and $\mathbf{P}_{adj,3}$ can be formulated as

$$\mathbf{P}(\xi) = \mathbf{P}_{adj,1} + \frac{\xi}{d}(\mathbf{P}_{adj,3} - \mathbf{P}_{adj,1}) + \alpha\xi(d - \xi)(\mathbf{P}_i - \mathbf{P}_m)$$

where $d$ is given by

$$d = \|\mathbf{P}_{adj,3} - \mathbf{P}_{adj,1}\|_2$$

and $\alpha$ will be determined later. The point $\mathbf{P}_m$ is defined by

$$\mathbf{P}_m = \mathbf{P}_{adj,1} + \beta(\mathbf{P}_{adj,3} - \mathbf{P}_{adj,1}).$$

If we insert this into the scalar product

$$(\mathbf{P}_i - \mathbf{P}_m) \cdot (\mathbf{P}_{adj,3} - \mathbf{P}_{adj,1}) = 0$$

we obtain

$$\beta = \frac{(\mathbf{P}_i - \mathbf{P}_{adj,1}) \cdot (\mathbf{P}_{adj,3} - \mathbf{P}_{adj,1})}{(\mathbf{P}_{adj,3} - \mathbf{P}_{adj,1})^2}.$$

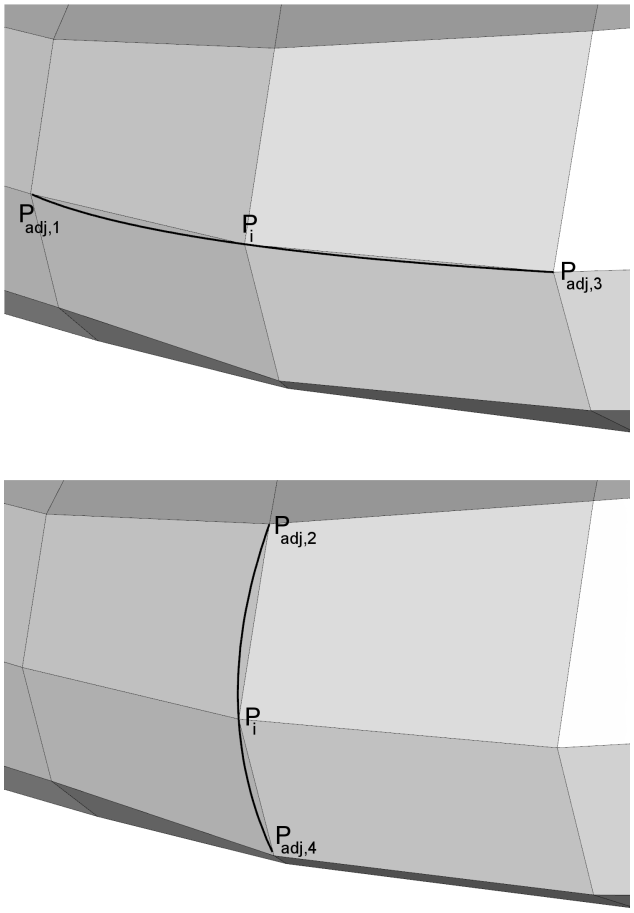


**Fig. 14.** Smoothing by moving control points along parabolae

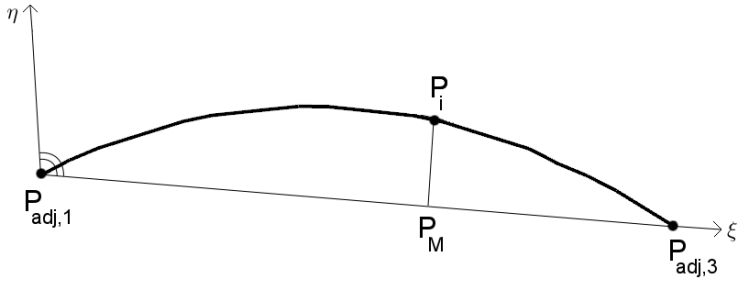


**Fig. 15.** Setting for the computation of a planar parabola passing through three points

Finally, we have to determine $\alpha$ in such a way that the parabola passes through $\mathbf{P}_i = \mathbf{P}(\beta d)$, leading to the equation

$$\mathbf{P}_i = \mathbf{P}_{adj,1} + \beta(\mathbf{P}_{adj,3} - \mathbf{P}_{adj,1}) + \alpha\beta d^2(1-\beta)\left(\mathbf{P}_i - \mathbf{P}_{adj,1} - \beta(\mathbf{P}_{adj,3} - \mathbf{P}_{adj,1})\right)$$

which is equivalent to

$$\alpha\beta d^2(1-\beta) = 1 \qquad \Leftrightarrow \qquad \alpha = \frac{1}{\beta d^2(1-\beta)}.$$

Now we can move $\mathbf{P}_i$ into the direction of $\mathbf{P}\left(\frac{1}{2}d\right)$.

The method only works well if $\mathbf{P}_{adj,1}$ or $\mathbf{P}_{adj,3}$ do not lie too close to $\mathbf{P}_i$ because the parabola may become very steep in that case. Moving the control point along such a steep parabola would increase the distance of that point from the mesh quickly. Hence, we constrain the method to values of $\beta$ between 0.3 and 0.7. If $\beta$ is smaller than 0.3 we move $\mathbf{P}_{adj,1}$ away from $\mathbf{P}_i$ on the line through $\mathbf{P}_{adj,1}$ and $\mathbf{P}_{adj,3}$ until $\beta$ becomes equal to or greater than 0.3 defining the new artificial point $\bar{\mathbf{P}}_{adj,1}$. Then the flattened parabola passing through $\bar{\mathbf{P}}_{adj,1}$, $\mathbf{P}_i$ and $\mathbf{P}_{adj,3}$ is computed. Analagously, we move $\mathbf{P}_{adj,3}$ away from $\mathbf{P}_i$ if $\beta$ is greater than 0.7 and construct the artificial point $\bar{\mathbf{P}}_{adj,3}$.

Finally, the control point is moved into the direction of the point $\mathbf{P}\left(\frac{1}{2}d\right)$ by using a damping factor. A good choice has turned out to be

$$\xi = \left(\frac{3}{4}\beta + \frac{1}{8}\right) d.$$

If the control point currently considered ($\mathbf{P}_i$) is an extraordinary vertex, i.e., having a valence of $N \neq 4$ and thus not being connected to four adjacent vertices by edges, we have to use another smoothing method. If $N = 3$ we compute the plane through the three adjacent control points connected to $\mathbf{P}_i$ by edges. Now we define a local coordinate system by using the unit vectors spanning the plane as $x$- and $y$-direction. Hence, the signed distance of a control point to the plane gives the $z$-coordinate. We then can fit an explicit surface

$$z(x,y) = ax^2 + bxy + cy^2 + dx + ey + f \qquad (\star\star)$$

to the control point $\mathbf{P}_i$ and its six adjacent control points (the three connected by edges as mentioned above and another three only connected by faces). If we translate the plane such that $\mathbf{P}_i$ lies in it we can choose $\mathbf{P}_i$ to be the origin of the local coordinate system. Hence, $f = 0$ forces the plane to pass through $\mathbf{P}_i$. The other five coefficients are computed by inserting the local coordinates of the six adjacent control points into $(\star\star)$ and solving the resulting approximation problem via a QR-algorithm. The new position for the control point $\mathbf{P}_i$ finally is determined by taking the $x$- and $y$-values of the barycenter of the triangle defined by the three adjacent control points connected by edges and inserting them into the equation for the fitted surface to obtain a $z$-coordinate.

If $N \geq 5$ the approach is similar: Instead of a plane we have to use a regression plane, since the control point $\mathbf{P}_i$ in this case has five or more adjacent control points connected by edges. To these and to the remaining adjacent control points being only connected by faces (at least five) we fit the surface

$$z(x, y) = ax^3 + bx^2y + cxy^2 + dy^3 + ex^2 + fxy + gy^2 + hx + iy + j.$$

Hence, we have to solve an approximation problem again.

**Example**

Figure 16 again shows a close-up of the approximated wing tip after the same number of iterations as in Fig. 13 without smoothing (top) and with two smoothing steps after each approximation (bottom) leading to a more uniform distribution of mesh points.

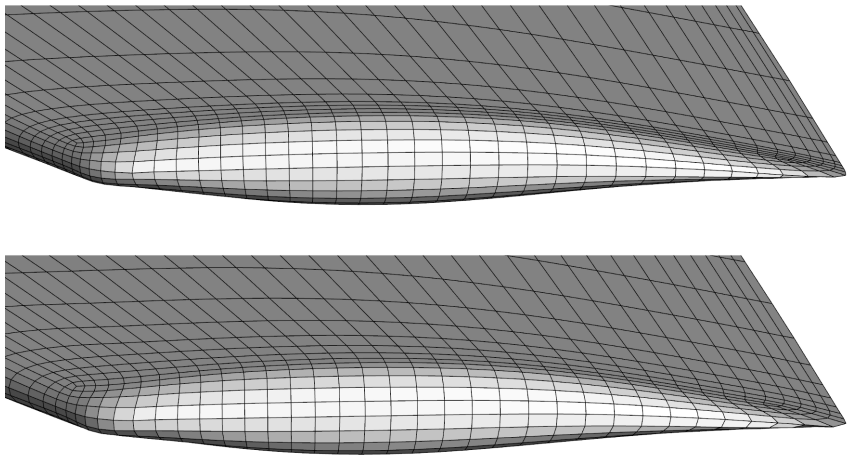

**Fig. 16.** Surface mesh (close-up of wing tip) after three process iterations, top: without smoothing, bottom: with two smoothing steps after each approximation

## 5 Conclusion and Future Work

We have presented a fast iterative procedure for the generation of high-quality block-structured volume meshes with an inner surface well conforming to a given target surface. Apart from the manual generation of two initial polyhedra the whole process operates automatically. It uses modified Catmull-Clark surface subdivision rules to produce smooth surface meshes, extended by the possibility to model creases, and tensor-product Catmull-Clark volume subdivision rules for the refinement of the flow field between the inner and the

outer surface. The points of the inner Catmull-Clark limit surface can be precomputed after each subdivision and then be projected onto a given B-spline surface by using the Nelder-Mead algorithm. New control points of the mesh can then be obtained by approximating the projected limit points. This linear least-squares problem is solved by applying the CGLS-method. After an approximation one or more smoothing steps can be applied to obtain a more uniform distribution of mesh control points.

We will evaluate the quality of our implementation by using the resulting volume meshes for numerical flow simulations as already done for simple geometries in [9]. From the sub-project *High Reynolds Number Aero-Structural Dynamics (HIRENASD)* of the SFB 401 *Flow Modulation and Fluid-Structure Interaction at Airplane Wings* at RWTH Aachen (see [12]) we have experimental data from wind tunnel readings for the wing in Fig. 9 attached to a simplified fuselage. Furthermore, from the follow-up project *Aero-Structural Dynamics Methods for Airplane Design (ASDMAD)* (see [13]), a cooperation with Airbus, we have additional experimental results for a wing with a winglet. We will compare the results for both configurations to the simulation results in order to have an indicator for the quality of the overall process.

## Acknowledgments

The first author is funded by the German Research School for Simulation Sciences (GRS), Jülich, Germany.

## References

1. J. Clark and E. Catmull. Recursively generated B-spline surfaces on arbitrary topological meshes. *CAD*, 10(6):350–355, 1978.
2. J. Stam. Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values. In *Proceedings of SIGGRAPH*, pages 395–404, 1998.
3. T. DeRose, M. Kass, and T. Truong. Subdivision Surfaces in Character Animation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM SIGGRAPH, pages 85–94, 1998.
4. J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.
5. K. Joy and R. MacCracken. The Refinement Rules for Catmull-Clark Solids. Technical Report CSE-96-1, Department of Computer Science, University of California, Davis, 1996.
6. C. Bajaj, S. Schaefer, J. Warren, and G. Xu. A subdivision scheme for hexahedral meshes. *The Visual Computer*, 18:343–356, 2002.
7. K.-H. Brakhage and Ph. Lamby. Application of B-Spline Techniques to the Modeling of Airplane Wings and Numerical Grid Generation. *CAGD*, 25(9):738–750, 2008.

8. K. H. Brakhage. Modified Catmull-Clark Methods for Modelling, Reparameterization and Grid Generation. In *Proceedings of the 2. Internationales Symposium Geometrisches Modellieren, Visualisieren und Bildverarbeitung*, HfT Stuttgart, pages 109–114, 2007.
9. M. Rom. Oberflächenreparametrisierung und Gittererzeugung für numerische Strömungssimulationen mit Hilfe von Catmull-Clark-Methoden. Diplomarbeit, RWTH Aachen, 2009.
10. C. de Boor. *A Practical Guide to Splines*. Springer, 1978.
11. Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2nd edition, 2003.
12. W. Schröder, editor. *Summary of Flow Modulation and Fluid-Structure Interaction Findings*. Notes on Numerical Fluid Mechanics and Multidisciplinary Design, Vol. 109. Springer, 2010.
13. J. Ballmann, A. Boucke, B.-H. Chen, L. Reimer, M. Behr, A. Dafnis, C. Buxel, S. Buesing, H.-G. Reimerdes, K.-H. Brakhage, H. Olivier, M. Kordt, J. Brink-Spalink, F. Theurich, and A. Büscher. Aero-Structural Wind Tunnel Experiments with Elastic Wing Models at High Reynolds Numbers (HIRENASD - ASDMAD). 49th AIAA Aerospace Sciences Meeting, Orlando, Florida, 2011.