Improved Mesh Deformation

Holger Barnewitz and Bernd Stickan

Abstract. An improved, robust, error reducing CFD-mesh deformation module for the parallel simulation environment FlowSimulator is presented. The mesh deformation method is based on radial basis function interpolation for the surface- and volume- mesh nodes combined with a group-weighting and displacement-blending approach. Since the latter weighting and blending approaches are based on given wall distances to the group surfaces, another module for the wall distance computation is introduced. Due to performance reasons, the number of input data locations (base points) used for the radial basis function interpolation must be limited. Therefore, methods have been developed to reduce the number of base points while keeping the interpolation error as low as possible. Furthermore, the modules have been parallelized for usage in multi-node high performance computing clusters. Finally, the capability of a multidisciplinary, parallel application is demonstrated in FlowSimulator with reduced errors and uncertainties.

1 Introduction

Airbus strategy to essentially move much more towards simulation makes it indispensable to know about any uncertainties and deficiencies in the predictive capabilities used for aerodynamic development. Knowing about error bands, their quantity and having in hand some means to manage and minimize their influence on the predicted results could tremendously help in the development process, reliable optimization of the product, shortening of development time and cost.

Holger Barnewitz · Bernd Stickan
AIRBUS Operations GmbH,
Airbus-Allee 1, D-28199 Bremen
e-mail: {holger.barnewitz,bernd.b.stickan}@airbus.com

The MUNA project is an essential brick within the Airbus strategy of flight physics/aerodynamics focusing on providing adequate tools for numerical qualification of aerodynamic design during concept phase. Respectively qualified CFD is expected to form the single basis for judgement of aerodynamic status before entering concentrated high level wind tunnel testing – to be ready for next new aircraft development. In addition, MUNA is contributing to support aerodynamic data process change towards "more simulation, less testing".

The contribution described in the following sections focuses on CFD mesh deformation used in the context of numerical aerodynamic shape optimization and shape design including static wing deformation. Major topics are:

- New mesh deformation module "FSDeformation" with advanced methods and integration into FlowSimulator [1]
- Geometry parametrization with a link between CAD (CATIA V5) and mesh deformation
- Use of mesh deformation for aerodynamic shape optimization
- Application of mesh deformation in a CFD/CSM coupled iterative process
- Combining shape design and CFD/CSM coupling in a multi-disciplinary optimization

A typical multi-disciplinary optimization (MDO) process chain for shape optimization of a wing including the static deformation is shown in Fig. 1. An essential brick is the mesh deformation tool which is applied to:

1. reflect the changed shape design generated by a parametric CAD model,
2. deform the wing according to aerodynamic (and other) forces.

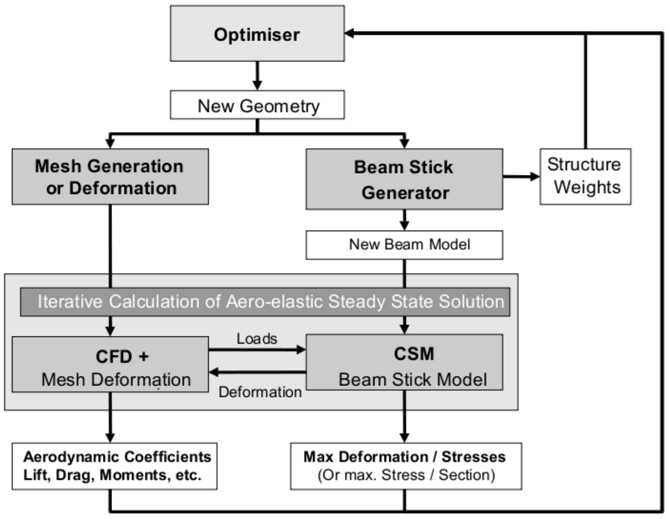

Optimiser
New Geometry
Mesh Generation or Deformation
Beam Stick Generator
Structure Weights
New Beam Model
Iterative Calculation of Aero-elastic Steady State Solution
CFD + Mesh Deformation
Loads
Deformation
CSM Beam Stick Model
Aerodynamic Coefficients Lift, Drag, Moments, etc.
Max Deformation / Stresses (Or max. Stress / Section)

Fig. 1 Multi-disciplinary optimization chain for wing shape and structure weight

The advantage of using mesh deformation for *unstructured grids* is manifold:

- It avoids the problem of *numerical noise* for the calculated aerodynamic coefficients which might occur if new meshes are created for slightly changed geometries. This "noise" is caused by the change of mesh topology. Mesh deformation conserves the topology and small geometry variations produce small mesh deformations in a continuous way.
- A so-called *restart* capability of the flow solver allows to start from a flow solution calculated beforehand to save computing time.
- Usually, deformation of an unstructured CFD mesh is faster than re-generating a new mesh, and thus also saves computing time.

Mesh deformation plays a key role in aerodynamic shape optimization, since any adjustment of the model geometry has an impact on the 3D CFD-mesh. Because CFD-simulations usually rely on spatial discretization based on volume-meshes, these have to be updated if a CAD surface changes its location or its shape. The costs for the generation of new meshes should be as low as possible but should also produce usable grids even for large local changes in the model geometry. There are several possible ways to update the mesh, e.g. the complete re-meshing of the complete grid or the deformation of an existing grid.

For unstructured meshes, re-meshing would certainly change the topology of the mesh. Since the discretization in 3D space is generally not dense enough to produce a mesh-independent flow solution, the newly generated mesh would produce a slightly different flow field not caused by the geometry change. This leads, especially in the case of shape optimization, to *noise in the aerodynamic coefficients*, which often significantly disturbs the shape optimizer. Furthermore, the computational cost for re-meshing of unstructured grids is very high. Hence, mesh deformation is an essential tool in this area of computational fluid dynamics:

- the topology of the mesh remains unchanged and
- small geometry variations correspond to small changes of the *numerically* determined aerodynamic flow field.

2 CFD Mesh Deformation Module

The deformation module "FSDeformation" has been developed for the simulation environment FlowSimulator [1]. It is based on an implementation of the mesh-deformation module presented in [6] using the radial basis function (RBF) interpolation approach. This approach is extended by the feature of specifying groups of different boundaries with separate interpolation functions and a blending function, which restricts the deformation to a specified zone around these boundaries. Both features are controlled by the distances $\boldsymbol{d}$ of the mesh nodes to the group target boundaries. The distances are calculated by the wall distance module (section 3).

More details about the deformation module, for example concerning parallel performance and interpolation quality, can be found in [10].

2.1 Radial Basis Functions in Mesh Deformation

Deformation methods based on RBF-interpolation are independent of the volume mesh and flow solver type, because the algorithm is working on completely arbitrary clouds of points without using any connectivity information. Additionally, for the mesh updates of consecutive optimization steps or an unsteady aeroelastic simulation, only a matrix-vector multiplication is necessary for each mesh node. The computationally most expensive part is to compute the interpolation matrix for this multiplication. It can be calculated once in the beginning of the entire simulation and remains unchanged, since it only depends on the base points, but not on the deformation vectors. Consequently, the method can be perfectly parallelized (using MPI and partitioned grids), because each process has to apply the interpolation matrix only to its own grid nodes. But it is also clear that the dimension of the interpolation matrix highly influences the overall speed of the algorithm.

2.1.1 Multivariate Interpolation Using Radial Basis Functions

The radial basis functions approach is a well-established interpolation method for gridded and scattered data, whereas the most natural context for function approximation is given for scattered data [5, p. 99], [4, p. 4]. In the field of computational fluid dynamics (CFD) it is often used for coupling CFD-grids to finite element structure grids.

The input data in d dimensions consist of data locations $\boldsymbol{x}_i$, merged into the dataset

$$X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n\} \in \mathbb{R}^d, \tag{1}$$

and the corresponding function values

$$f_i = f(\boldsymbol{x}_i) \in \mathbb{R}, \quad i = 1, ..., n. \tag{2}$$

The data locations $\boldsymbol{x}_i$ are called centers or *"base points"*.

The goal is to interpolate the function values between the base points by an approximant $s : \mathbb{R}^d \to \mathbb{R}$ to satisfy the condition

$$s|_X = f|_X \,. \tag{3}$$

In this specific case s is a linear combination of shifted radially symmetric basis functions ϕ. Radially symmetric means that the value of $\phi\,(\cdot)$ depends only on the distance of the argument to the origin, hence it is often written $\phi\,(\|\cdot\|)$. The distance norm is usually the Euclidean norm (with $d = 3$)

$$\|\boldsymbol{x}\|_2 = \sqrt{\sum_{i=1}^{d} x_i^2}\,. \tag{4}$$

$s(\boldsymbol{x})$ has the general form

$$s(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \phi\left(\|\boldsymbol{x} - \boldsymbol{x}_i\|\right). \tag{5}$$

Setting $s(\boldsymbol{x}_i)$ equal to f_i for all $i \in \{1,..,n\}$ leads to the linear system

$$A\boldsymbol{y} = \boldsymbol{b} \tag{6}$$

with

$$A = \left(\phi\left(\|\boldsymbol{x}_j - \boldsymbol{x}_k\|\right)\right)_{(j,k)=1,..,n}, \quad \boldsymbol{y} = (\alpha_i)_{i=1,..,n}, \quad \boldsymbol{b} = (f_i)_{i=1,..,n}. \tag{7}$$

A unique interpolant is usually (for most ϕ) guaranteed, if the base points are all distinct and there are at least two of them [3, p. 6]. An example for a radial basis function could be $\phi\left(\|\boldsymbol{x}\|\right) = \|\boldsymbol{x}\|^2 \log\|\boldsymbol{x}\|$, which is called *"thin plate spline"*.

An important attribute of this interpolation method is the possibility to expand the approach of equation (5) by adding a polynomial to the definition without losing the uniqueness of the coefficients. For function values f_i, which show a polynomial character, the appended polynomial improves the interpolation quality. The only restriction is that the polynomial must have a degree $m \geq 1$ and is non-zero at all base points. This leads to:

$$s(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \phi\left(\|\boldsymbol{x} - \boldsymbol{x}_i\|\right) + p(\boldsymbol{x}). \tag{8}$$

The coefficients can be computed by solving

$$s(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \phi\left(\|\boldsymbol{x} - \boldsymbol{x}_i\|\right) + p(\boldsymbol{x}) = f_i \tag{9a}$$

$$0 = \sum_{i=1}^{n} \alpha_i q\left(\boldsymbol{x}_i\right) \ \forall\, q: \ \mathrm{deg(q)} \leq \mathrm{deg(p)} \tag{9b}$$

The extra equation (9b) takes up the extra degrees of freedom given by the polynomial coefficients, to allow a unique interpolant. The uniqueness can be guaranteed, if ϕ is "conditionally positive definite". It is referred to [5, p. 101] for more details to the theory of this topic.

Again, the requirements on X are not very strong. For a linear polynomial, X must only contain four base points, which do not lie on a plane. Furthermore, if the function values f_i at the base points were generated by a linear function, they would be reproduced exactly by the linear polynomial. [4, p. 5]

In the following the dimension is set to $d = 3$ in this document. Since $\boldsymbol{x} = (x_x, x_y, x_z)$, the polynomial is linear and can be written as

$$p(\boldsymbol{x}) = \beta_0 + \beta_1 x_x + \beta_2 x_y + \beta_3 x_z = \boldsymbol{\beta}^T \begin{pmatrix} 1 \\ \boldsymbol{x} \end{pmatrix}. \tag{10}$$

So equations (9) can be abstracted to matrix notations

$$H\boldsymbol{y} = \boldsymbol{b}, \tag{11}$$

with

$$A = \left(\phi\left(\|\boldsymbol{x}_j - \boldsymbol{x}_i\|\right)\right)_{(j,i)=1,..,n} \quad \in \mathbb{R}^{n\times n}, \tag{12}$$

$$P = \left(\begin{pmatrix} 1 \\ \boldsymbol{x}_k \end{pmatrix}_{k=1,..,n}\right) \quad \in \mathbb{R}^{4\times n}, \tag{13}$$

$$H = \begin{pmatrix} 0 & P \\ P^T & A \end{pmatrix} \quad \in \mathbb{R}^{(n+4)\times(n+4)}, \tag{14}$$

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{\beta} \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} (\beta_i)_{i=1,..,4} \\ (\alpha_i)_{i=1,..,n} \end{pmatrix} \quad \in \mathbb{R}^{n+4} \text{ and} \tag{15}$$

$$\boldsymbol{b} = \begin{pmatrix} 0 \\ \boldsymbol{f} \end{pmatrix} = \begin{pmatrix} 0 \\ (f_i)_{i=1,..,n} \end{pmatrix} \quad \in \mathbb{R}^{n+4}. \tag{16}$$

Solving (11) provides in $\boldsymbol{b}$ the coefficients to use (8) for the interpolation of arbitrary points. The matrix H will be called *"interpolation matrix"* below, although it is only used to calculate the interpolation coefficients.

The module presented is not independent of cell connectivity, since wall distances of the nodes to certain boundary groups are used. The algorithm to compute the wall distances relies on connectivity information to determine neighboring nodes. But, it is important to note that the base points $\boldsymbol{x}_{s,i}$, $i = 1,..,n_\mathrm{s}$ do not need any connectivity information.

Solutions for the indicated performance factor "interpolation matrix size", which directly depends on the number of used input deformation vectors, will be shown in section 4. That section contains different methods to reduce the number of base points and deformation vectors.

The basic interpolation functions of the module are taken from DLR's flow solver TAU. They have been applied successfully at DLR and Airbus to many test cases.

2.2 Algorithm

The interpolation algorithm is based on a *group-weighting* and a *deformation-blending* approach.

A group-weighting approach is used to allow the independent movement of different model parts/boundaries in the grid. Otherwise the deformations of different boundaries could influence each other and unintentional surface deformation would be the result. Separating the interpolation by group protects the shape of the different bodies. Therefore, the interpolation matrix H^g of each group g has to be computed and applied to the grid nodes separately. Finally, the deformation result for each grid point is calculated by a weighted average of each group-deformation result.

The deformation-blending approach supports the protection of boundary layer cells and the usage of radial basis functions $\phi(\|x\|)$ with limits $\phi(\|x\|) \to \infty$ for $\|x\| \to \infty$. These radial basis functions, which increase with increasing distance to the base point of a deforming body, need to be restricted farther away from the surface of this body. Otherwise local deformations would influence the whole mesh. Additionally, the added polynomial of the interpolation approach (8) would deform the whole volume mesh as well. Consequently, this approach that is implemented to recover linear deformations exactly, cannot be used without the blending of deformation values.

Hence, the notations are expanded by an elevated group index g for n_g groups. As input data there are n_{s}^g base points $\boldsymbol{x}_{s,i}^g \in \mathbb{R}^3$ for each group g merged into the datasets

$$X_{\mathrm{s}}^g = \left\{ \boldsymbol{x}_{s,1}^g, \boldsymbol{x}_{s,2}^g, \ldots, \boldsymbol{x}_{s,n_{\mathrm{s}}^g}^g \right\} \quad \text{for } g = 1,..,n_g. \tag{17}$$

The function values that are going to be interpolated, are the deformation vectors

$$\Delta \boldsymbol{x}_{s,i}^g = \Delta \boldsymbol{x}\left(\boldsymbol{x}_{s,i}^g\right) = \begin{pmatrix} \Delta x_{s,i}^{g,x} \\ \Delta x_{s,i}^{g,y} \\ \Delta x_{s,i}^{g,z} \end{pmatrix} \in \mathbb{R}^3 \;\; \text{for } i = 1,\ldots,n_{\mathrm{s}}^g, \;\; g = 1,..,n_g, \tag{18}$$

which could be used to compute the displaced coordinates $\boldsymbol{x}_{\mathrm{new,i}}^g$ of the base points:

$$\boldsymbol{x}_{\mathrm{new,i}}^g = \boldsymbol{x}_{s,i}^g + \Delta \boldsymbol{x}_{s,i}^g \quad \text{for } \; i = 1,\ldots,n_{\mathrm{s}}^g, \;\; g = 1,..,n_g. \tag{19}$$

But the aim of the deformation module is to update the mesh nodes and not the base points.

A difference to the function values f_i in equation (2) to the function values $\Delta \boldsymbol{x}_{s,i}^g$ is their dimension. Section 2.1.1 only deals with one-dimensional function values while in this case the function values are three-dimensional. Therefore each coordinate of the mesh nodes has to be interpolated separately. It is advantageous that the interpolation matrix H^g in (11) has to be computed only once for each boundary group instead of computing it for each dimension separately, since the matrix depends only on the base points $\boldsymbol{x}_{s,i}$ and the chosen radial basis function ϕ. So the interpolation matrices H^g for each group can be stated as:

$$H^g = H\left(X_{\mathrm{s}}^g, \phi\right). \tag{20}$$

For each dimension $k \in \{x,y,z\}$ the interpolation coefficients $\boldsymbol{\alpha}^{g,k} = \left(\alpha_i^{g,k}\right)_{i=1,..,n_{\mathrm{s}}^g}$ and $\boldsymbol{\beta}^{g,k} = \left(\beta_i^{g,k}\right)_{i=1,..,4}$ can be calculated by inverting equation (11):

$$\begin{pmatrix} \boldsymbol{\beta}^{g,k} \\ \boldsymbol{\alpha}^{g,k} \end{pmatrix} = \left(H^g\right)^{-1} \begin{pmatrix} \mathbf{0} \\ \left(\Delta \boldsymbol{x}_{s,i}^{g,k}\right)_{i=1,..,n_{\mathrm{s}}^g} \end{pmatrix}. \tag{21}$$

The actual interpolation algorithm calculates the deformations of the grid nodes

$$\boldsymbol{dx}_{\mathrm{v,i}} = \left(dx^x_{\mathrm{v,i}} \; dx^y_{\mathrm{v,i}} \; dx^z_{\mathrm{v,i}} \right)^T \tag{22}$$

for the volume mesh grid nodes $\boldsymbol{x}_{\mathrm{v,i}}$ by using the distance d_i^g to the nearest surface of group g. For every coordinate $k \in \{x,y,z\}$ the governing equations are:

$$dx^{g,k}_{\mathrm{v,i}} = \sum_{j=1}^{n_{\mathrm{s}}^g} \alpha_j^{g,k} \phi \left(\|\boldsymbol{x}_{\mathrm{v,i}} - \boldsymbol{x}_{s,j}\| \right) + \left(\boldsymbol{\beta}^{g,k} \right)^T \begin{pmatrix} 1 \\ \boldsymbol{x}_{\mathrm{v,i}} \end{pmatrix}, \; g = 1,..,n_g \tag{23}$$

$$\mathrm{blend}(\mathrm{d_i^g}, \mathrm{g}) = \begin{cases} 0 & : d_i^g > \mathrm{RZW^g} \\ 1 & : d_i^g < \mathrm{RFW^g} \\ \frac{\mathrm{RZW^g} - \mathrm{d_i^g}}{\mathrm{RZW^g} - \mathrm{RFW^g}} & : \quad \text{else} \end{cases} \tag{24}$$

$$\mathrm{weight}(\mathrm{d_i^g}) = \frac{1}{\sqrt{\max\{d_i^g, \varepsilon\}}} \tag{25}$$

$$dx^k_{\mathrm{v,i}} = \frac{\sum_{g=1}^{n_g} \mathrm{blend}(\mathrm{d_i^g}, \mathrm{g}) \cdot \mathrm{weight}(\mathrm{d_i^g}) \cdot \mathrm{dx^{g,k}_{v,i}}}{\sum_{g=1}^{n_g} \mathrm{weight}(\mathrm{d_i^g})} \tag{26}$$

Two new functions have been introduced: the blending function blend$(\cdot)$ and the weighting function weight$(\cdot)$. The weighting function averages the individual group deformations. Because its limit for $d_i \to 0$ is infinity, it needs a cut-off value $1/\sqrt{\varepsilon}$ for numerical reasons.

The blending function is sketched in figure 2. With its group-parameters $\mathrm{RZW^g}$ (Radius Zero Weight) and $\mathrm{RFW^g}$ (Radius Full Weight) it is controlling the deformation of the grid nodes. If a grid node is close to a boundary of group g with a distance less than $\mathrm{RFW^g}$, it will move approximately like the boundary. This functionality can be used to conserve the sensitive boundary layer part of a grid. Farther away from the boundary with a distance $d_i^g > \mathrm{RZW^g}$ the deformation is zero.

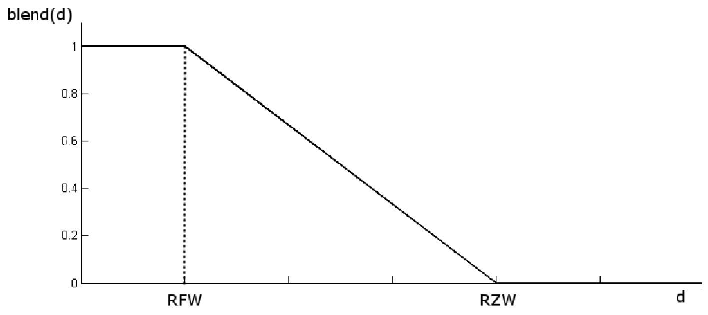


Fig. 2 Blending function for grid node deformation computation, including the parameter radius full weight (RFW) and radius zero weight (RZW)

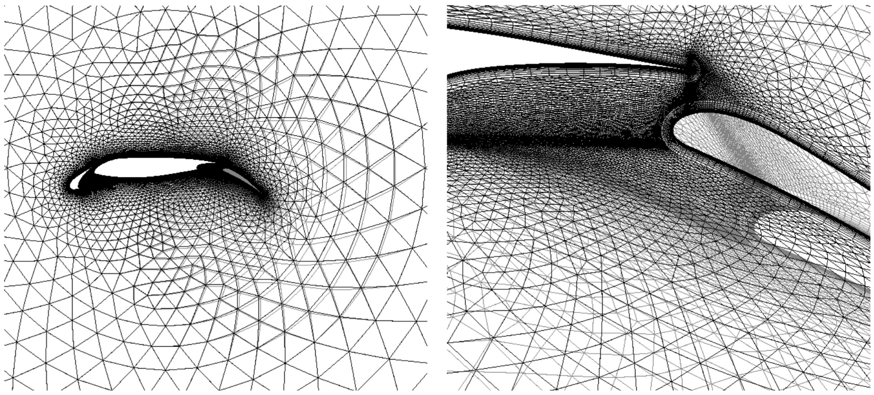

(a) Overall view (b) Zoom view

Fig. 3 2d test case, wing including flap and slat. Each of the 3 parts is an independent deformation group and only the flap has input values unequal to zero (undeformed: black, deformed: grey)

An example for independently deforming groups can be seen in figure 3. It shows that the surface mesh of the rigid main wing body is not affected by the deformation of the nearby moving flap. The radius zero weight can be recognized in figure 3(a), too.

The algorithm is also described shortly in [6]. This paper also provides test cases showing the usefulness of the presented group-weighting approach and the quality conserving capability of the methodology.

Several different boundary type dependent algorithms have been developed to simplify the usage of standard cases often applied to CFD meshes for aircrafts:

- *Standard Boundary Type.* This is handled as described above. Deformation vectors have to be provided for this surface type.
- *No-Normal-Movement Boundary Type.* All surface points on this boundary are allowed to slide on the surface. The movement in surface normal direction is suppressed. It's used for example for symmetry planes.
- *Far-field Boundary Type.* Here the deformation is set to zero.
- *Attached Group Boundary Type.* This treatment conserves the shape of an attached device, e.g. an engine mounted on a deforming wing.

3 Wall Distance Module

The mesh deformation module presented in section 2 uses the distance of the grid points to the closest group boundary for the weighting of groups and for the blending of deformations. The wall distances control the influence of different boundary groups on the deformation of a specific volume grid node.

There are different approaches for the computation of wall distances, for example based on partial differential equations, as seen in [11], based on a clever merging of the boundary points [13], or just a naive brute force algorithm, which compares each boundary node with every volume mesh node. The presented method, which was adopted from DLR's TAU preprocessor, uses another algorithm. In TAU the wall distance is used for the application of certain turbulence models. The method is, like the mesh deformation module, embedded into an independent module for the simulation environment FlowSimulator.

The algorithm uses an advancing frontier approach. Every grid node n gets an additional parameter $\boldsymbol{x}_{\text{near}}[n]$, which saves the coordinates of the currently nearest boundary node. Then in every iteration step, each node compares the distance to its $\boldsymbol{x}_{\text{near}}$-entry with the $\boldsymbol{x}_{\text{near}}$ values of its neighboring nodes and where required updates the $\boldsymbol{x}_{\text{near}}$-value with a better value from a neighbor. Since the boundary nodes have the correct solution directly at the beginning, the solution for $\boldsymbol{x}_{\text{near}}$ for each node moves into the field node by node.

This so-called advancing front algorithm makes it possible that for certain (structured) grids it would take only a few iteration steps to advance the correct solution for $\boldsymbol{x}_{\text{near}}$ into the interior of the grid.

4 Base Point Reduction Methods

The number of base points n_{s} has a major influence on the performance of the radial basis function interpolation algorithm. The needed (direct) matrix inversion depends on the third power of n_{s} and the interpolation of the grid points depends linearly on the base point number. If the tool is used for the coupling of a structural finite elements (FEM) grid to a computational fluid dynamics grid, the number of input base points will be equal to the number of surface grid nodes of the FEM-grid. The common number of surface nodes of these grids is way too large to use them all for the RBF grid deformation and still having satisfactory runtime results. So the reduction of the base points is indispensable for the mesh deformation module.

The reduction of the base points is not the only way to increase the efficiency of radial basis function interpolation methods. Other possibilities are, for example, multilevel approaches combined with base point reduction [9] or partition of unity approaches like in [12]. The multilevel approach uses a base point set hierarchy to start the interpolation at a coarse level and then refining it progressively. The partition of unity approach breaks the large problem down to several small ones by partitioning the base points into neighbor sets.

A useful attribute of the radial basis function interpolation approach is that no connectivity information of the input base points is needed. To conserve this characteristic the reduction algorithms do not use connectivity information as well.

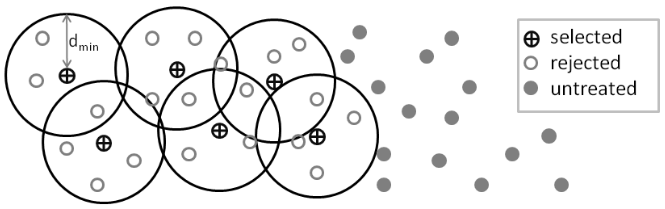


Fig. 4 Equidistant reduction snapshot during iteration step

4.1 Equidistant Reduction Method

The TAU deformation module also contains this method to reduce the number of base points. It tries to select the base points X_s spatial-evenly distributed from the point set X_inp. This is managed by iteratively finding the right minimal distance d_min to possible neighboring base points, to get as close as possible to the maximal number of desired base points $n_\mathrm{s,max}$. Neighbors with a distances less than d_min are rejected during this process. Due to performance issues, it is using an octree data structure to find the neighbors closer than d_min to a specified base point. Figure 4 sketches one iteration step of the algorithm.

The result is having evenly distributed base points. But choosing the base points like this does not take into account the deformation vectors $\Delta x_\mathrm{inp,i}$ or *interpolation errors*.

4.2 Weighted Distances

This approach is similar to the equidistant reduction approach of section 4.1. The idea is to modify the distances between two input sites $x_\mathrm{inp,i}$ and $x_\mathrm{inp,j}$ by a weighting factor $w_{i,j}$. The consequence would be an increased point density in areas of higher weights.

The distance $d_{i,j}$ between two points x_i and x_j with associated weights w_i and w_j is calculated by

$$d_{i,j} = \frac{w_i + w_j}{2} \|x_i - x_j\|_2. \tag{27}$$

The disadvantage of this approach is that the octree data structure of section 4.1 cannot be used any more. Instead a simple list data structure has to be used. Searching for neighbors of one node will therefore include to check the distance to all input sites X_inp.

4.2.1 Weighting by the Difference to the Local Average Deformation

This weighting approach takes the deformation vectors $\Delta x_{\mathrm{inp,i}}$ into account directly. It uses a function $\mathit{FindNeighbours}(\boldsymbol{x}, X, \Delta X, d)$, which returns subsets of

$$X_{\mathrm{nb}} = \left\{x_{\mathrm{nb,1}}, x_{\mathrm{nb,2}}, \ldots, x_{\mathrm{nb,n_{nb}}}\right\} \tag{28}$$

and

$$\Delta X_{\mathrm{nb}} = \left\{\Delta x_{\mathrm{nb,1}}, \Delta x_{\mathrm{nb,2}}, \ldots, \Delta x_{\mathrm{nb,n_{nb}}}\right\} \tag{29}$$

of cardinality n_{nb} with points of distance less than d to $\boldsymbol{x}$. In this case the distances are not weighted yet. The weights for the input point $x_{\mathrm{inp,i}}$ are then calculated by

$$(X_{\mathrm{nb}}, \Delta X_{\mathrm{nb}}) = \mathit{FindNeighbours}\left(x_{\mathrm{inp,i}}, X_{\mathrm{inp}}, \Delta X_{\mathrm{inp}}, d\right) \tag{30}$$

$$\overline{\Delta x_{\mathrm{nb}}} = \frac{\sum_{i=1}^{n_{\mathrm{nb}}} \|\Delta x_{\mathrm{nb,i}}\|_2}{n_{\mathrm{nb}}} \tag{31}$$

$$w_i = \frac{\left|\overline{\Delta x_{\mathrm{nb}}} - \|\Delta x_{\mathrm{inp,i}}\|_2\right|}{\sum_{i=1}^{n_{\mathrm{inp}}} \left|\overline{\Delta x_{\mathrm{nb}}} - \|\Delta x_{\mathrm{inp,i}}\|_2\right|} \tag{32}$$

The idea is to develop an expression that favors the base points, the absolute deformation value of which is different to the average deformation value $\overline{\Delta x_{\mathrm{nb}}}$ of its neighbors. Another thought is that the nodes at the outer tips of a deforming body will get a higher weight, since at the tip the deformations reach usually their maximum and consequently differ strongly from the neighborhood mean. An approach which only takes the gradient into account would not result in a higher weight for the outer base points, because the deformation gradient would not have a peak at an outer base point. An example to illustrate this idea can be seen in figure 5. This simplified example shows why the approach leads to higher weights for the base points on the tip, the deformation vectors of which should not be neglected in the final base point set. It shows a tip body with a slight rotational deformation. The deformation vector with the largest value is on the tip of the body. The equidistant reduction algorithm from section 4.1 could easily fail to select the maximum deformation vectors. Because of the higher weight values at the tip this would happen less likely with the new algorithm.

This example shows a disadvantage of the algorithm as well. If the gradient of the deformation vectors is constant in a certain area, the weights will tend to zero. This will lead to a very low base point density in the next step. To get a lower border for the density, the final reduction method is combined with the equidistant reduction method. First a fraction $\mathrm{frac}_{\mathrm{equi}}$ of the desired $n_{\mathrm{s,max}}$ base points is chosen by the equidistant algorithm, then the remaining base points and deformation vectors are selected with the weighted distance approach.

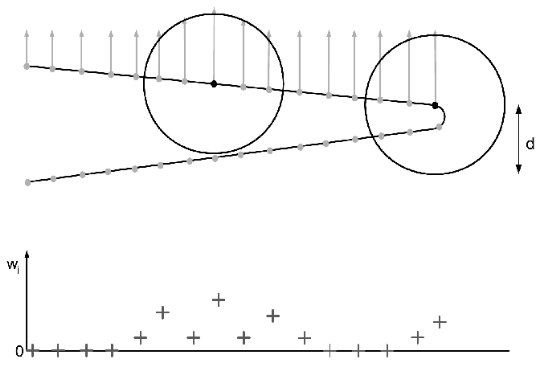


Fig. 5 Schematic example for weights w_i of the upper base points with their deformation vectors

4.2.2 Weighting by Interpolation Error

The algorithm presented in this section is combining the approach of the weighted distance reduction with the interpolation error calculated for the input data locations. In this case, the distance weights

$$W = \{w_1, w_2, ..., w_{\text{inp}}\} \tag{33}$$

are equal to the error of interpolated deformation vectors $\Delta\widetilde{X}_{\text{inp}}$.

The basic scheme of the algorithm looks like:

- Select start base point set X_s with corresponding deformation vectors ΔX_s by equidistant reduction
- Do n_{EWSteps} times:
  - Interpolate deformations at input points X_{inp}, by using the sets X_s and ΔX_s, to get the deformation vectors $\Delta\widetilde{X}_{\text{inp}}$
  - Calculate weights w_i by comparing ΔX_{inp} to $\Delta\widetilde{X}_{\text{inp}}$
  - Add further base points and deformation vectors by weighted distance reduction to X_s and ΔX_s, respectively.

The type of greedy algorithm, which recalculates the exact interpolation error in each step, is also proposed in [2, p.9].

To interpolate the input points X_{inp} in each step, a new interpolation matrix H_k has to be created from the already chosen base points X_s and inverted in every iteration step. Then the error can be calculated by interpolating the deformation vectors ΔX_s of these base points to the input set X_{inp} to get the interpolated data set

$$\Delta\widetilde{X}_{\text{inp}} = \left\{\Delta\widetilde{\boldsymbol{x}}_{\text{inp},1}, \Delta\widetilde{\boldsymbol{x}}_{\text{inp},2}, ..., \Delta\widetilde{\boldsymbol{x}}_{\text{inp},n_{\text{inp}}}\right\} \tag{34}$$

and taking the pairwise difference to the input deformation vector set ΔX_{inp} to compute the weights

$$w_i = \|\Delta\widetilde{\boldsymbol{x}}_{\text{inp,i}} - \Delta x_{\text{inp,i}}\|_2, \quad i = 1,2,...,n_{\text{inp}}. \tag{35}$$

4.3 Error Correction

The error correction algorithm was originally presented in [2, p. 7]. The algorithm tries to correct the $\boldsymbol{\alpha}$-interpolation coefficient vector locally. Therefore, unlike the previous methods, it is not using the interpolation approach including a polynomial (9), but instead the basic approach without an added polynomial (5):

$$s(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \phi\left(\|\boldsymbol{x} - \boldsymbol{x}_i\|\right). \tag{36}$$

Furthermore, the coefficients α_i are not calculated by inverting the interpolation matrix A, but instead by correcting them during the iterations continuously. In each step the interpolation error $e_i, i = 1,..,n_{\text{inp}}$ of all deformation vectors

$$\Delta\widetilde{X} = \left\{\Delta\widetilde{\boldsymbol{x}}_1, \Delta\widetilde{\boldsymbol{x}}_2, ..., \Delta\widetilde{\boldsymbol{x}}_{n_{\text{inp}}}\right\} \tag{37}$$

at the data sites

$$X_{\text{inp}} = \left\{\boldsymbol{x}_{\text{inp},1}, \boldsymbol{x}_{\text{inp},2}, \ldots, \boldsymbol{x}_{\text{inp},\text{n}_{\text{inp}}}\right\} \tag{38}$$

is recalculated. The coefficients

$$\boldsymbol{\alpha} = (\alpha_i)_{i=1,..,n_{\text{inp}}} \tag{39}$$

and the deformation vectors $\Delta\widetilde{\boldsymbol{x}}_i$ are adjusted locally by the radial basis function belonging to the base point with the largest interpolation error $e_{i_{\text{worst}}} = \|\mathrm{d}\Delta\boldsymbol{x}_{\text{i}_{\text{worst}}}\|_2$. The correction of $\alpha_{i_{\text{worst}}}$ is performed by

$$\Delta\alpha_{i_{\text{worst}}} = \frac{1}{\phi(0)} \mathrm{d}\Delta\tilde{\boldsymbol{x}}_{\text{i}_{\text{worst}}}, \tag{40}$$

which is used to update the interpolation values of all base points by

$$\Delta\tilde{\boldsymbol{x}}_i = \Delta\tilde{\boldsymbol{x}}_i + \Delta\alpha_{i_{\text{worst}}} \phi\left(\|\boldsymbol{x}_i - \boldsymbol{x}_{i_{\text{worst}}}\|_2\right) \tag{41}$$

In every step in equation (41) the error $e_{i_{\text{worst}}}$ at of the deformation $\Delta\widetilde{\boldsymbol{x}}_{i_{\text{worst}}}$ is changed to zero, since

$$\begin{aligned} \Delta\widetilde{\boldsymbol{x}}_{i_{\text{worst}}} &= \Delta\widetilde{\boldsymbol{x}}_{i_{\text{worst}}} + \frac{\phi(\|\boldsymbol{x}_{i_{\text{worst}}} - \boldsymbol{x}_{i_{\text{worst}}}\|_2)}{\phi(0)} \mathrm{d}\Delta\tilde{\boldsymbol{x}}_{\text{i}_{\text{worst}}} \\ &= \Delta\widetilde{\boldsymbol{x}}_{i_{\text{worst}}} + \Delta\boldsymbol{x}_{i_{\text{worst}}} - \Delta\widetilde{\boldsymbol{x}}_{i_{\text{worst}}} = \Delta\boldsymbol{x}_{i_{\text{worst}}}. \end{aligned} \tag{42}$$

But the corrections for the deformation vectors $\Delta\tilde{\boldsymbol{x}}_i$, which are located inside the impact area of the base point $\boldsymbol{x}_{i_{\text{worst}}}$, are not necessarily decreasing the interpolation error. Hence, if $n_{\text{s,max}} > n_{\text{inp}}$ base points should be selected, the algorithm will run infinitely without reducing the interpolation error to zero. Additionally, the algorithm "tends to show a degree of self limiting behavior in terms of how many points it uses ([...]), often returning to correct a previously identified point rather than introducing a new one" [2, p. 8].

The paper [2] uses the algorithm above to approximate the coefficients α_i of equation (36), but also recommends not to use these coefficients. Instead the selected base points in X_{s} should be used for exact interpolation, which uses the inversion of the interpolation matrix as seen in section 2.1.1. The algorithm implemented into the deformation module presented in this document uses this recommendation and, secondly, instead of the interpolation approach (36) the approach including a polynomial as seen in equation (8) for the interpolation matrix creation.

Because the results were still not satisfactory, this approach has been combined with an initial equidistant reduction step to choose $\text{frac}_{\text{equi}} \cdot \text{n}_{\text{s,max}}$ base points by the algorithm presented in section 4.1.

A big disadvantage of the algorithm is that it only works with radial basis functions $\phi(r)$ with the maximal value for $r = 0$, so radial basis functions with local influence range.

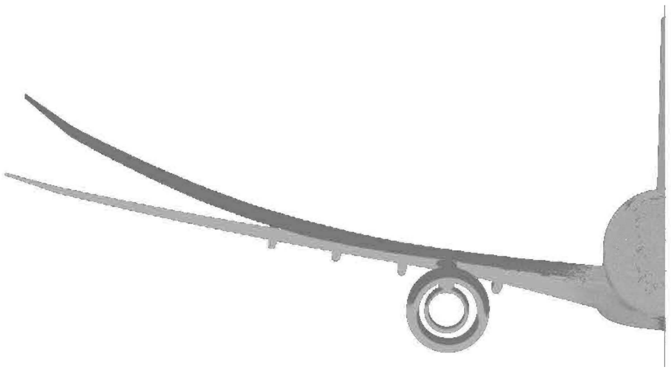


Fig. 6 Half model test case

5 Interpolation Quality Comparison

The different reduction algorithms in section 4.1 to 4.3 select different base point sets X_s from the input data site set $X_{\boldsymbol{inp}}$. This section is comparing the resulting interpolation errors in a test case.

Therefore the extremely deformed half model airplane, as seen in figure 6 is used. The input base points X_{inp} and their deformation vectors ΔX_{inp} were calculated with a structural loads program. The tool generates for each surface grid node a deformation vector, so the cardinality of X_{inp} and ΔX_{inp} is quite large with a value of $n_{\text{inp}} = 137,136$ for the wing without engine only.

Figure 7 shows the interpolation error e_i for the bottom surface of the wing, because it used to show higher interpolation errors. The settings for the interpolation and base point reduction can be seen in table 1.

Because the structural loads tool gives an deformation output for every surface grid node, the interpolation error for the surface nodes $\boldsymbol{e} = (e_i)_{i=1,..,n_{\text{inp}}}$ can be calculated by taking the differences between the calculated interpolations $\Delta\widetilde{\boldsymbol{x}}_i$ and the input deformations $\Delta\boldsymbol{x}_i$:

$$e_i = \|\Delta\widetilde{\boldsymbol{x}}_i - \Delta\boldsymbol{x}_i\|_2. \tag{43}$$

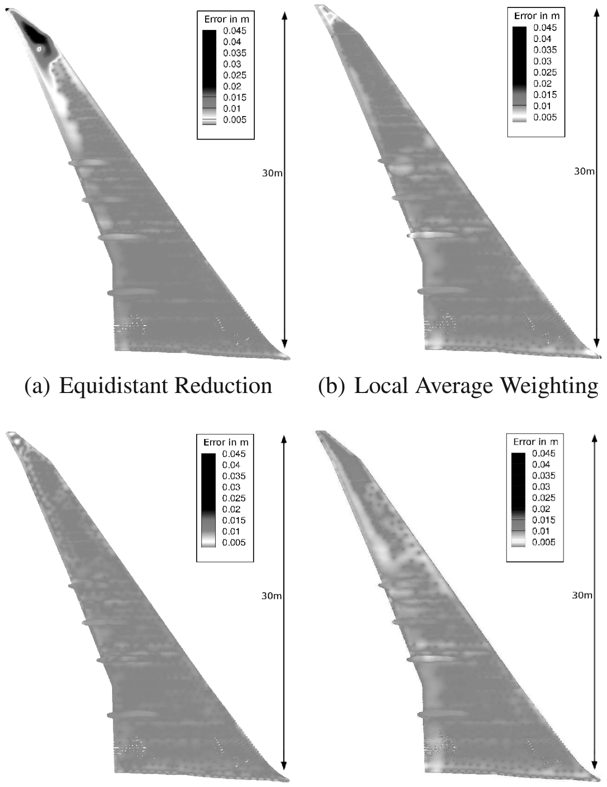
The picture 7(a) clearly shows that the base points, chosen by the equidistant reduction method, are not satisfactory in the outer wing part. The outer 30 percent of the wing show strongly increased error values. This result has motivated to improve the base point selection process. The other reduction algorithms show a strongly improved interpolation error in this part of the wing, too. The mean absolute error $\overline{\boldsymbol{e}}$ of each test case for the actual 2000 base point setting, and additionally for another test series with only 1000 base points is given in table 2. Furthermore, this table contains the variance $Var(\boldsymbol{e})$ and the maximal error $\max_i(e_i)$.

The table confirms the impression of the given plots: All new methods choose base points resulting in a significantly lower interpolation error. Furthermore, the variance $Var(\cdot)$ indicates that less fluctuations in the error can be expected. The maximum error is lowered by up to 92 percent.

Figure 8 shows how the base points are chosen by the different algorithms. The equidistant reduction algorithm (7(a)) distributes the base points nicely over the whole domain. Taking a closer look, the decreased density of points in the thin parts of the wing, like the trailing edge and the tip, can be recognized.

Table 1 Test settings

Reduction method	Parameter	RBF ϕ
Equidistant reduction	-	Wendland's C^0, impact radius $r = 20.0$
Local average weighting	$\text{frac}_{\text{equi}} = 0.5$, $d_{\text{min}} = d_{\text{max}}/10$	
Error weighting	$\text{frac}_{\text{equi}} = 0.5$, $n_{\text{EWSteps}} = 3$	
Error correction	$\text{frac}_{\text{equi}} = 0.5$	



(a) Equidistant Reduction (b) Local Average Weighting

(c) Error Weighting (d) Error Correction

Fig. 7 Interpolation error of wing, lower surface view, 2000 base points, color table: absolute interpolation error in m

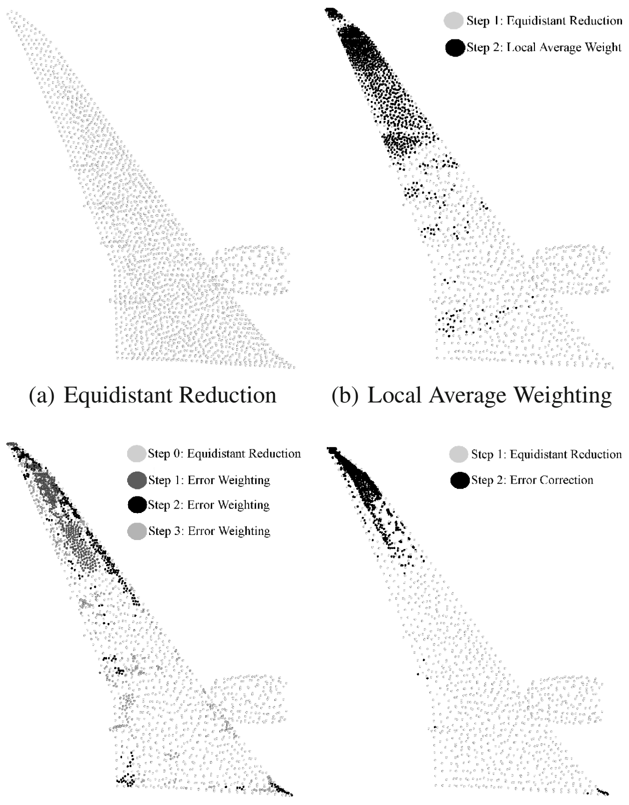
Table 2 Test results interpolation error $\boldsymbol{e} = (e_i)_{i=1,..,n_{\text{inp}}}$

(a) 1000 base points

Reduction method	$\overline{\boldsymbol{e}}\ [m]$	$Var(\boldsymbol{e})\ [m^2]$	$\max_i(e_i)\ [m]$
Equidistant reduction	5.30E-03	8.52E-02	1.24E-04
Local average weighting	1.56E-03	3.87E-02	3.83E-06
Error weighting	1.52E-03	1.10E-02	3.00E-06
Error correction	1.40E-03	8.99E-03	1.19E-06

(b) 2000 base points

Reduction method	$\overline{\boldsymbol{e}}\ [m]$	$Var(\boldsymbol{e})\ [m^2]$	$\max_i(e_i)\ [m]$
Equidistant reduction	2.33E-03	2.52E-05	4.99E-02
Local average weighting	8.60E-04	1.92E-06	1.91E-02
Error weighting	6.86E-04	6.20E-07	7.92E-03
Error correction	7.38E-04	4.55E-07	3.83E-03


Step 1: Equidistant Reduction
Step 2: Local Average Weight
Step 0: Equidistant Reduction
Step 1: Error Weighting
Step 2: Error Weighting
Step 3: Error Weighting
Step 1: Equidistant Reduction
Step 2: Error Correction

(a) Equidistant Reduction (b) Local Average Weighting

(c) Error Weighting (d) Error Correction

Fig. 8 2000 base points selected by different reduction algorithms

The new methods have all used the equidistant reduction algorithm in the first step for half of their base points. The remaining half has been selected differently, besides all algorithms concentrate the selected points in the outer wing part.

The local average method (8(b)) is the most extreme example for this behavior. Because the deformations are increasing with a parabolic character, the weights used to be more significant in the outer part. Supplementary, the more narrow getting wing supports this behavior, because the neighborhood of a certain point would contain more points of the side closer to the fuselage then from the outer side, which influences the local mean deformation.

The two remaining approaches based on the interpolation error are choosing their base points similarly. A difference between the error based greedy algorithms is that the error weighting algorithm is distributing the points more numerous in areas far away from the outer wing. The error correction algorithm has selected most points in the tip area.

The results show the best point selection for the error weighting and error correction method. But one has to keep in mind that error correction is only working with radial basis functions with limited influence range, while error weighting has the larger computational costs (7 times larger than error correction).

6 Applications

6.1 Wing Shape Design

To accurately compare the aerodynamic coefficients (e.g. drag) between small changes of the aerodynamic shape of a wing, it is necessary to be as independent from the CFD mesh as possible. Otherwise, difficulties arise to distinguish between grid discretization effects ("numerical noise" due to change of grid topology) and geometric effects. Nowadays in industrial context a 3D *unstructured* CFD mesh is not made in a way to obtain a *mesh independent* CFD solution.

However, mesh deformation conserves the grid topology and small geometry variations produce small mesh deformations in a continuous way. Utilizing this, comparisons of aerodynamic coefficients are better possible and thus uncertainties otherwise introduced by changes of grid topology are minimized. Mesh deformation with FSDeformation was here successfully applied to a shape design change for a wing-tip (figure 9).

The discrete deformation field was obtained from the parametric CAD model (CATIA V5) using a two-stage process (first the treatment of curves and then surfaces). In a predictor step, the deformation field is determined by subtracting points on discretized corresponding CAD curves. This gives an initial surface deformation which may not be accurate on the inner region of surface panels apart from the bounding curves.

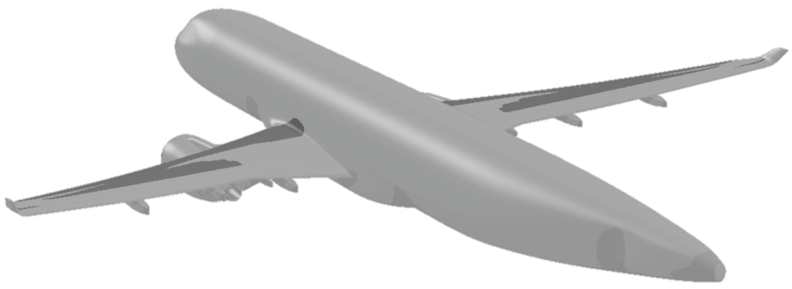
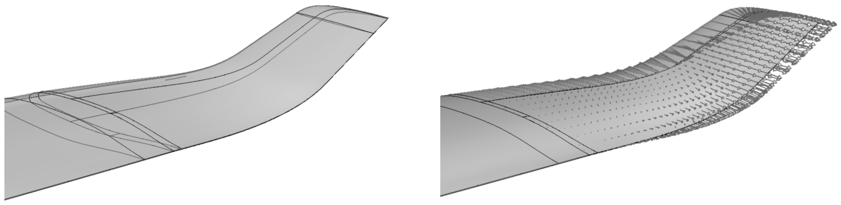


Fig. 9 Wing tip shape design example



(a) Parametric CAD geometry (b) Discrete deformation field calculated from parametric CAD

Fig. 10 Wing tip design with deformation obtained from a parametric CAD geometry

In the corrector step surface points are projected to the new CAD geometry. The projection vectors together with the displacement vectors of the discretized curves then builds the final discrete deformation field (figure 10(b)). It serves as input for the new tool FSDeformation to accurately move all points of the CFD grid corresponding to the parametric change of the CAD geometry.

6.2 Multi-disciplinary Wing Optimization–SFB-401-Wing

The task given was to optimize a wing with respect to aerodynamics, structures, and performance under considerations of static aeroelastic effects. The study involves the sizing of the wing box skins and spars to obtain minimum weight fulfilling static aero-elastic requirements (details in [7, p. 287]).

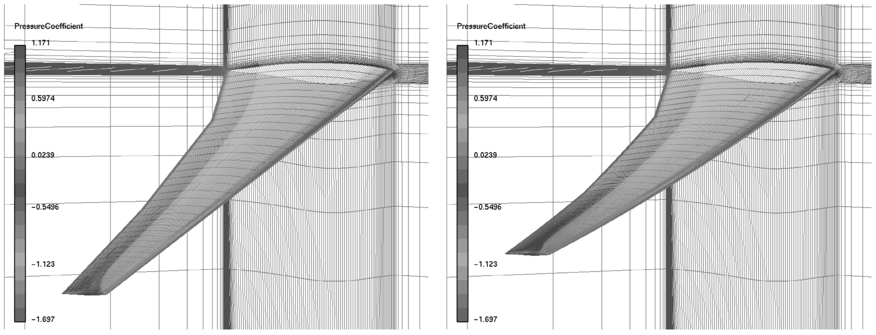
The considered MDO process chain for shape optimization of a wing including the static deformation is shown schematically in Fig. 1. The objective function is:

$$\mathrm{Obj} = W_{\mathrm{A/C}} \times C_{\mathrm{D}}/C_{\mathrm{L}}, \tag{44}$$

where $W_{\mathrm{A/C}}$ is the total weight of the aircraft, C_{D} is the overall aerodynamic drag coefficient, and C_{L} is the aerodynamic lift coefficient. The objective, thrust, is equivalent to the total aerodynamic drag force in stationary horizontal flight, which should be minimized.

A CATIA V5 parametric model of the wing is controlled by the optimizer using an external CATIA-DesignTable, where all relevant shape parameters for the wing are listed. The shape of airfoils at four predefined wing sections (root, kink1, kink2, and tip section) can be changed parametrically to control the thickness, camber, and twist distribution of the wing. The wing planform is fixed.

Two structure design parameters control the *relative thickness change* of the wing front and rear spars in combination with the upper and lower sheet thicknesses of the



(a) Step 1, no deformation (b) Step 10, aeroelastic equilibrium

Fig. 11 Undeformed and deformed wing with pressure coefficient distribution and CFD mesh for the CFD/CSM coupled iterative process.

wing box. The stiffness and the weight of the wing are depending on these structure parameters.

Mesh deformation is a crucial component here. On the one hand, the change in geometry shape design through the parametric CAD model (CATIA V5) is treated by mesh deformation, on the other hand, the deformation of the wing structure depending on the aerodynamic forces (in addition to other forces such as fuel weight, engines, etc.) is also covered by applying mesh deformation. The CFD/CSM coupling is displayed in figure 11.

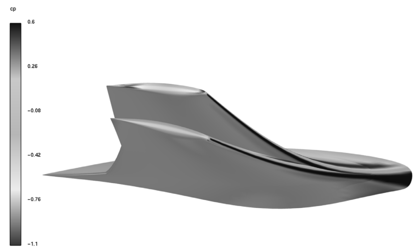
The individual components of the process chain were used in the parallel, in-memory FlowSimulator environment [1], so that a time-consuming and data intensive exchange of files was not required. Compared to the former methods, which used file exchange, significant time savings of around 50 percent have been obtained. This a major step forward in an industrial context together with the accuracy improvements and reductions of uncertainties.

In figure 12(a) and 12(b) the results of the optimization are presented. Shown is the original geometry in the aeroelastic equilibrium and the optimized geometry with a significantly different twist distribution and bending.

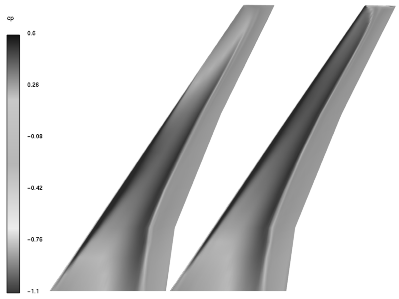
Figure 13 shows the convergence of the required thrust during the optimization process using a gradient free Downhill Simplex optimizer [8]. After around 80 design changes the optimum has been reached nearly.

6.3 Application to Complex Configuration

It was found that FSDeformation in the parallel FlowSimulator environment could be applied successfully to very complex, industrially relevant problems. An example for a coupled CFD/CSM application for an complete aircraft in high-lift configuration with deflected flaps and slats is shown in Fig. 14.



(a) Side view to check change of bend and twist. Lower wing: optimized, Upper wing: initial



(b) Top view to check change of pressure coefficient distribution. Left: optimized, Right: initial

Fig. 12 Clean wing CFD-CSM optimization

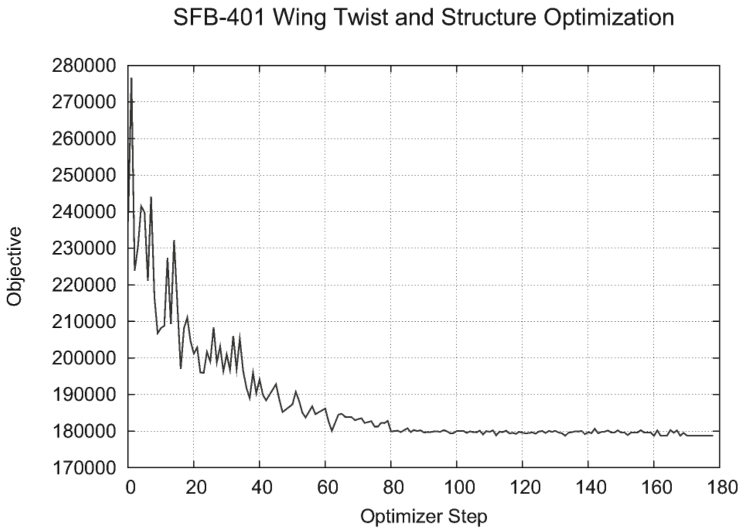


Fig. 13 Clean wing CFD-CSM optimization. Convergence of the required thrust in horizontal stationary flight during the optimization process.

7 Summary

This work has presented a mesh deformation module for the parallel simulation environment FlowSimulator. The module is based on the radial basis function interpolation approach.

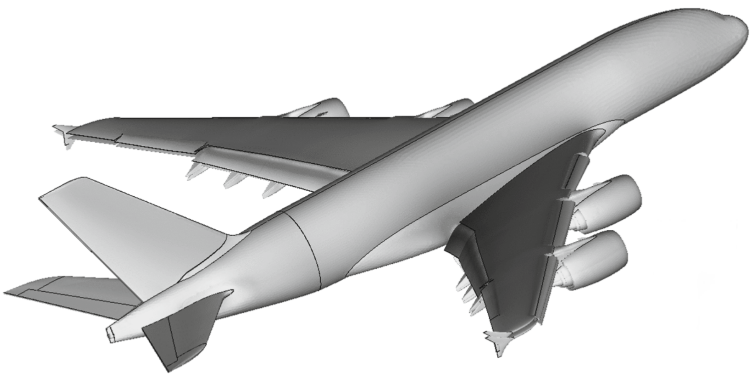


Fig. 14 Complex high-lift configuration for a CFD/CSM coupled simulation applying the new tool FSDeformation. Shown is the deformed and non-deformed geometry (see wing tip) due to a different aerodynamic load case.

The application fields of mesh deformation is manifold (unsteady simulations, shape optimization and aeroelasticity). In a next step, radial basis function interpolation and the mesh deformation module and its algorithm have been introduced. The module combines the radial basis function interpolation approach with a group-weighting and deformation-blending feature. This allows to move different surface groups/bodies independently from each other. Furthermore, the deformation blending provides an improved protection of boundary layer cells and allows the usage of unbounded radial basis functions and the extension of the radial basis function interpolation approach by a polynomial.

The group-weighting and deformation-blending uses the wall distance of the volume mesh nodes to the group boundaries. Hence a wall distance computation module has been implemented. It uses an advancing-front algorithm for the distance computation.

Additionally, the deformation module was extended with new deformation group features. These features allow to define deformation groups without creating base points and deformation vectors for this group. The features support far-field boundaries, symmetry planes or the rigid attachment of a boundary group to another boundary group.

Because the computational cost of the interpolation algorithm depends on the number of interpolation base points, the module offers four different methods for the reduction of the input base points. The first method uses an octree data-structure to select equidistant base points. The remaining three methods use in the first step this method as well. But in the second step they either use weighted distances for a modified equidistant reduction function, or they correct the interpolation error locally by selecting base points individually. The two weighted distance reduction methods use the difference to the mean deformation of the neighboring nodes or the interpolation error of the base points not selected for the weight computation. All the different methods have been compared in terms of performance and interpolation error. Here the locally operating error correction method has shown very good results in performance/interpolation error efficiency. But the method is limited to locally supported radial basis functions. Because the radial basis function Euclid's Hat, which tends to infinity for an increasing input argument, has produced the lowest interpolation error, the interpolation error weighting method is the best advice.

The wall distance module and the deformation module have been parallelized with MPI. The theoretically perfect speedup of the interpolation method may only be affected by unbalanced node distributions over the parallel MPI processes.

The creation of the module FSDeformation by Airbus and its integration into the parallel FlowSimulator software environment has helped to reduce the uncertainties that occur in small geometry changes, which typically occur at aerodynamic shape design. Numerical shape optimization of components by using the improved grid deformation technique for unstructured grids has improved, reducing uncertainties related to mesh dependencies of the numerically obtained flow solutions.

Finally, applications for the deformation module in cooperation with the flow solver TAU and a structure module has been demonstrated. It has shown that the deformation module can play a key position in future computation chains using the simulation environment FlowSimulator with a perfect speedup for the mesh deformation method. Additionally, the example has illustrated that the coupling of different programs by using FlowSimulator can minimize intensive and time-consuming file input/output operations, due to the fact that both tools use the same main memory address space. The computational time and hence the cost of an optimization has been reduced considerably.

References

[1] FlowSimulator: Common simulation environment for integrated parallel CFD applications, `http://dev.as.dlr.de/gf/project/fsdm`

[2] Allen, C.B., Rendall, T.C.S.: Efficient mesh motion using radial basis functions with data reduction algorithms. In: AIAA Paper 2008-305, Department of Aerospace Engineering, University of Bristol, Bristol, Great Britain (January 2008)

[3] Baxter, B.J.C.: The Interpolation Theory. PhD thesis, Trinity College. University of Cambridge (August 1992)

[4] Beckert, A., Wendland, H.: Multivariate interpolation for fluid-structure-interaction problems using radial basis functions. Technical report, Institute of Aeroelasticity, German Aerospace Center (DLR), Institute for Applied Mathematics, University of Göttingen (March 2000)

[5] Buhmann, M.D.: Radial Basis Functions. Cambridge University Press (April 2003)

[6] Heinrich, R., Kroll, N., Neumann, J., Nagel, B.: Fluid-Structure Coupling for Aerodynamic Analysis and Design - A DLR Perspective. In: 46th AIAA Aerospace Sciences Meeting and Exhibit 2008, Reno (2008)

[7] Kroll, N., Schwamborn, D., Becker, K., Rieger, H., Thiele, F.: MEGADESIGN and MegaOpt - German Initiatives for Aerodynamic Simulation and Optimization in Aircraft Design: Results of the Closing Symposium of the MEGADESIGN and MegaOpt Projects. Springer (2007)

[8] Nelder, J.A., Mead, R.: A simplex method for function minimization. Computer Journal 7, 308 (1965)

[9] Ohtake, Y., Belyaev, A., Seidel, H.: Multi-scale and adaptive cs-rbfs for shape reconstruction from cloud of points (2003)

[10] Stickan, B.: Implementation and extension of a mesh deformation module for the parallel FlowSimulator software environment. Diploma Thesis at Airbus Deutschland GmbH, Bremen and Chair for Computational Analysis of Technical Systems, RWTH Aachen (2009)

[11] Tucker, P.G.: Differential equation-based wall distance computation for DES and RANS. PhD thesis, Department of Engineering, Fluid Dynamics Research Centre, University of Warwick (October 2002)

[12] Wendland, H.: Fast evaluation of radial basis functions: Methods based on partition of unity. In: Approximation Theory X: Wavelets, Splines, and Applications, pp. 473–483. Vanderbilt University Press (2002)

[13] Wigton, L.B.: Optimizing cfd codes and algorithms for use on cray computers, frontiers of computational fluid dynamics. World Scientific Publishing Co. Pte. (1998)