ADVANCED CFD-CSD COUPLING: GENERALIZED, HIGH PERFORMANT, RADIAL BASIS FUNCTION BASED VOLUME MESH DEFORMATION ALGORITHM FOR STRUCTURED, UNSTRUCTURED, AND OVERLAPPING MESHES

Matthias Schuff, Patrick Kranzinger, Manuel Keßler, and Ewald Krämer
kranzinger@iag.uni-stuttgart.de, University of Stuttgart, IAG, Pfaffenwaldring 21, Stuttgart, 70569, Germany

Abstract

Radial basis functions (RBF) are used for interpolation of moving 3D multigrid volume meshes based on their surface deformation. The method is independent of grid connectivity and can easily handle any structured, unstructured and overlapping mesh setup. For flexibility and reusability, an object oriented C++ library has been developed for easy integration into any flow solver. Furthermore, highly distributed computation is addressed. Since high resolution meshes contain way more surface points than accurate mesh deformation based on RBF interpolation needs, an efficient grid coarsening algorithm based on an octree has been implemented by which performance can be boosted considerably. Various setups using the new algorithm are presented. The method is applied to a typical dynamic aeroelastic problem – the fast-forward flight of a weakly coupled five bladed isolated rotor simulation. The validation was executed by comparing the resulting trim angles and loads with a formerly used algorithm.

NOMENCLATURE

α	Coefficient(s) for specific RBF
β	Coefficient(s) for specific polynomial
$\mathbf{M}_{bb}$	Matrix containing the evaluation of the RBF between each surface point
$\mathbf{M}_{complete}$	Full matrix needed for coeffcient computation
$\phi(\mathbf{x},\mathbf{d})$	Radial basis function with x as the evaluation point and d as a point of reference
φ	Azimuthal position of rotor blade
c_p	Pressure coefficient
d_b	Displacement of specific surface point(s)
l_{edge}	Edge length of an Octree cube
$l_{rootedge}$	Edge length of the root Octree cube
m,n	Factor for surface reduction
n_{level}	Octree level
n_{surf}	Amount of surface points
$n_{volumemesh}$	Amount of volume grid nodes
p	Arbitrary grid node(s)
P_b	Polynomial ansatz vector(s) for rigid body motion
p_{surf}	Surface point(s)
$q(\mathbf{p})$	Linear polynomial
$s(\mathbf{p})$	Displacement of a arbitrary grid node p

1 INTRODUCTION

Any motion of elastic, lift generating structures significantly influence the resulting aerodynamic loads. Because of the elasticity of helicopter rotor blades, including fluid-structure interaction for reproduction of the aeroelastic response is absolutely necessary when simulating helicopter aerodynamics.

For isolated rotor and complete helicopter simulations a weak coupling scheme between the structured Finite-Volume flow solver FLOWer[1] and a structure and flight mechanics code like HOST[2] or CAMRAD II[3] is used[4]. The elastic blade is represented by an Euler-Bernoulli beam discretized by rigid elements connected by virtual hinges. Modal shapes of the periodic deformations are passed to the CFD code defining the deformation of the surface for each azimuthal blade position. Up to now, based on these modal shapes within every CFD timestep the surfaces of the elastic rotor blades are reconstructed within FLOWer. Subsequently, the volume mesh is deformed using Hermite interpolation which requires connectivity information of the nodes of the structured mesh. Generally, this implementation requires structured single or multi block meshes, whose blocks have at least one point in common with the elastic surface. To a certain limit this can be bypassed by block splitting, so that blocks with connectivity to the surface are deformed at first. Afterwards the computed deformation of these blocks is applied to faces of the next, farther outside blocks. Considering block splitting on a larger scale, this procedure may become arbitrary complex and hence, a deformation order for the blocks is not feasible any more.

Within the last years the performance of a single computing core was nearly stagnating. Present day's high performance computation clusters offer instead 100,000 cores and more. Thus, efficiently parallelization is a key goal for being able to use the power of current and future computation clusters. Due to the restrictions to the rotor blade mesh topology, using the former implementation block splitting for an efficient parallelization was highly limited. Furthermore, blocks remained, whose geometric dimensions cannot be further shrunk. Increasing the spatial discretization would lead to higher number of grid nodes per block. Hence, the power of next generations of super computer cannot be used adequately.

To overcome these limitations and because of the ongoing development of the unstructured Discontinuous Galerkin code SUNWinT[5] and a coupled free wake method, a most flexible and reusable mesh deformation algorithm based on radial basis functions was developed, which is able to deform overlapping Chimera structures, unstructured meshes and structures with relative motion (e.g. slotted leading and trailing edge flaps). Furthermore, the algorithm can also represent any kind of hinge, by what connecting elastic to non-elastic components become possible.

2 DEFORMATION ALGORITHM

2.1 Radial basis function interpolation

Radial basis functions (RBF) can be used for interpolation of discrete data in an n-dimensional space. The deformation of a 3D-CFD mesh can thus be interpreted as an interpolation of the known discrete deformation of the surface in the surrounding area.

RBFs are real-valued functions whose value only depend on a certain distance[6], so that

$$\phi(\mathbf{x},\mathbf{d}) = \phi(\|\mathbf{x}-\mathbf{d}\|), \tag{1}$$

where x is the evaluation point and d the point of reference. These functions can be used for interpolating data by defining a function for each data point and summing them up. When deforming volume meshes the data point is a point on a surface and the data to be interpolated is its displacement in the deformed state. By adding a constant and linear polynomial to this sum, offsets and linear parts of the data are represented directly analytically. When handling CFD meshes, translation and rotation in space can be interpreted as x,y,z-offset and a linear function of x,y,z.

The displacement of an arbitrary point $\mathbf{p}$ may be described as

$$s(\mathbf{p}) = \sum_{i=1}^{n}\Big(\alpha_i\phi\left(\|\mathbf{p}-\mathbf{p}_{surf_i}\|\right)\Big) + q(\mathbf{p}). \tag{2}$$

where α_i is a weighting factor for each RBF and $q(\mathbf{p})$ is an arbitrary linear polynomial[7]:

$$q(\mathbf{p}) = \begin{pmatrix}\beta_0\\ \beta_1\\ \beta_2\\ \beta_3\end{pmatrix}\cdot\begin{pmatrix}1\\ \mathbf{p}_x\\ \mathbf{p}_y\\ \mathbf{p}_z\end{pmatrix}. \tag{3}$$

The requirement is that there is an n-sized amount of reference points - or *surface points* - p_{surf}, that have a known deformation.

Thus, the displacement of these surface points must exactly map with the $s(p_{surf})$ so that

$$s(p_{surf_i}) = d_{b_i} \tag{4}$$

with d_{b_i} being the known displacement of the deformed surface points.

As averaged rigid body motions should be solely covered by the polynomial part $q(\mathbf{p})$, the requirement

$$\sum_{i=1}^{n}\alpha_i\mathbf{P}_{b_i} = 0 \tag{5}$$

with

$$\mathbf{P}_{b_i} = \Big[1\ p_{surf_{ix}}\ p_{surf_{iy}}\ p_{surf_{iz}}\Big] \tag{6}$$

has to be fulfilled[8].

Eventually, an equation system can be built and the coefficients for the basis functions (α) and the rigid body motions (β) can be computed by inverting the resulting dense matrix $M_{complete}$:

$$\begin{bmatrix} d_b\\ 0\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{M}_{bb} & \mathbf{P}_b\\ \mathbf{P}_b^T & 0\end{bmatrix}}_{M_{complete}}\begin{bmatrix}\alpha\\ \beta\end{bmatrix} \tag{7}$$

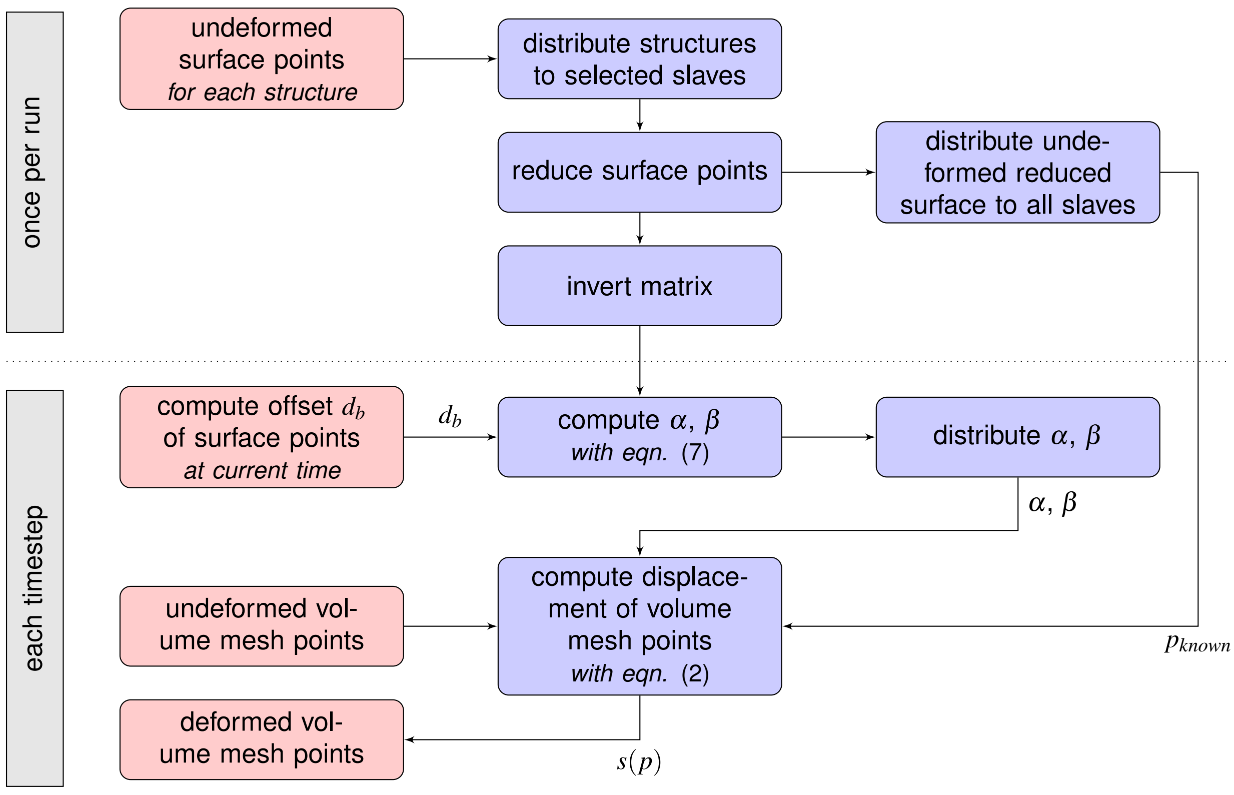

once per run
undeformed surface points for each structure
distribute structures to selected slaves
reduce surface points
distribute undeformed reduced surface to all slaves
invert matrix
each timestep
compute offset d_b of surface points at current time
d_b
compute α, β with eqn. (7)
distribute α, β
α, β
undeformed volume mesh points
compute displacement of volume mesh points with eqn. (2)
p_{known}
deformed volume mesh points
$s(p)$

Fig. 1. Flow chart of the RBF interpolation algorithm

where $\mathbf{M}_{bb}$ with row i and column j is the evaluation of the basis functions for the reference points between each other $\phi(p_{surf_i}, p_{surf_j})$, $\mathbf{P}_b$ is a $n \times 4$ matrix with the row i given as in (6).

As $M_{complete}$ contains no information of the actual displacement, but only information of the distances of each node to every other in non-deformed state, its inversion needs only to be executed once as long as the non-deformed surfaces won't change.

Detailed description of RBFs can be found in[6]. A detailed study of different RBFs for deforming volume meshes has been done by[8].

The performance of a RBF based deformation algorithm depends on two things. Firstly, the inversion of a densely populated $(n_{surf}+4) \times (n_{surf}+4)$ matrix. As a rule, this is an $(n_{surf}+4)^3$ problem and can be solved by LU decomposition or Gaussian elimination. Secondly, applying each RBF coefficient to each volume grid node, whereby the number of radial basis function coefficients is equal to the number of selected surface mesh points. Hence, the complexity for each timestep is $(n_{surf} \cdot n_{volumemesh})$. As n_{surf} can be kept constant (as shown in section 3) the resulting deformation algorithm complexity is $(n_{volumemesh})$.

The here presented algorithm only depends on knowing the displacement of reference points, what will typically be provided for the elastic surface by a CSD tool. No connection information between volume mesh nodes is needed and thus, the algorithm can be used for structured, unstructured and overlapping meshes alike.

2.2 Implementation

To make reusing the algorithm as easy and flexible as possible and to solve the problem of maintaining the interfaces to structure codes of each aerodynamic code, the new deformation algorithm is implemented as an object oriented library, which can easily be linked to any aerodynamic solver that can handle deformed meshes.

The process can be split into three major parts:

1. Setup of the surface points and their deformation (based on arbitrary deformation information scheme), leading to d_b (these objects shall be called *Deformable Objects*)
2. Computation of the RBF coefficients α and β with (7)
3. Deformation of the volume mesh, using (2)

In a common setup using Chimera technique, each rotor and its corresponding volume mesh blocks will

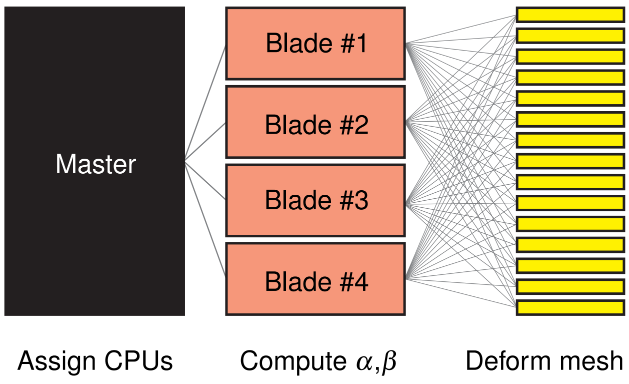


Fig. 2. Distribution of RBF coefficient computation and mesh deformation

be deformed independendly from the other blades. More sophisticated setups are of course thinkable and will be discussed later in this article. Thus, step 1 and 2 have to be carried out once for each structure that is deformed likewise, e.g. a rotor blade. The actual mesh deformation is highly distributable and can be done on the CPU where the aerodynamic solution is computed. Only the coefficients and the original surface points have to be distributed to each process, e.g. via MPI (see figure 2).

Furthermore, the steps are split into an initial part and a timestep part. Initial steps have to be executed once per run and include the setup of the surface points, a reduction of them and the inversion of the resulting matrix. The timestep part carries out the deformation of the surface points, the computation/distribution of the coefficients and the displacement of the volume mesh nodes, resulting in the deformed mesh.

Figure 1 shows the major processing of the algorithm as a flow chart.

Keeping the implementation rather simple, as a basis function the Volume Spline is used, which has been presented by[7,9]. The radial basis function

$$\phi(\mathbf{x},\mathbf{d}) = \|\mathbf{x}-\mathbf{d}\| \tag{8}$$

is the Euclidean distance itself. The farer the evaluation point is away from the center, the bigger is the influence. This smoothes the effect of local distortions[7].

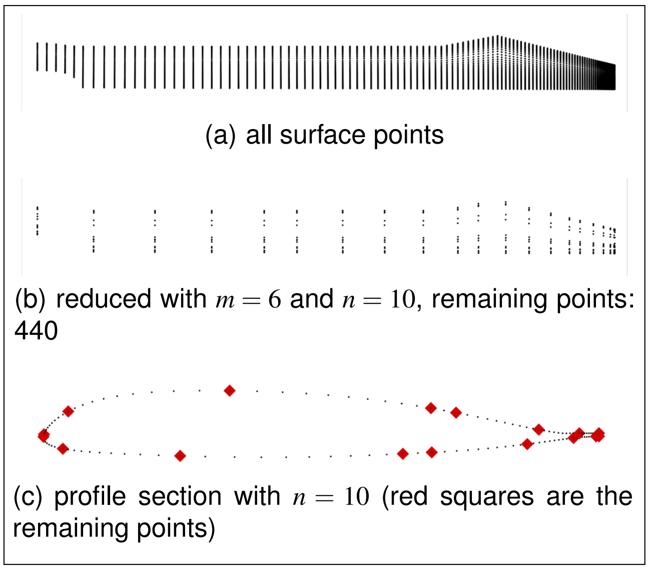
3 PERFORMANCE AND SURFACE REDUCTION

As mentioned in subsection 2.1, the performance of the mesh deformation depends strongly on the number of surface points used. Especially when using a very high resolution of the volume mesh, a high number of surface points will occur. It is shown by[10] and verified by our own experiences, that for RBF interpolation a lower amount than actually given is sufficient for an adequate mesh deformation. Furthermore, surface reduction algorithms have been explored by[10].

3.1 Reduction with simple method

For testing purposes, we firstly used a rather simple method, which is only applicable for structured meshes: The non deformed surface is extracted from the CFD mesh by collecting the block faces representing walls. For structural meshes this dataset consists of two-dimensional so-called *patches* of grid nodes, whose connectivity is well known. Each patch will be reduced by just taking every m-th point in the first and every n-th point in the second direction.

As an example, a structured rotor blade mesh with 19,919 surface points is used. The surface has been deformed using a fast-forward flight case at an azimuth of $\varphi = 270°$ with the tool HOST[2]. While creating the meshes, the orientation is chosen in such a manner that m represents the spanwise discretization and n runs around the airfoil. Figure 3(a) shows a view from above with all 19,919 points and figure 3(b) and 3(c) shows after a reduction to only 440 remaining surface points.



(a) all surface points

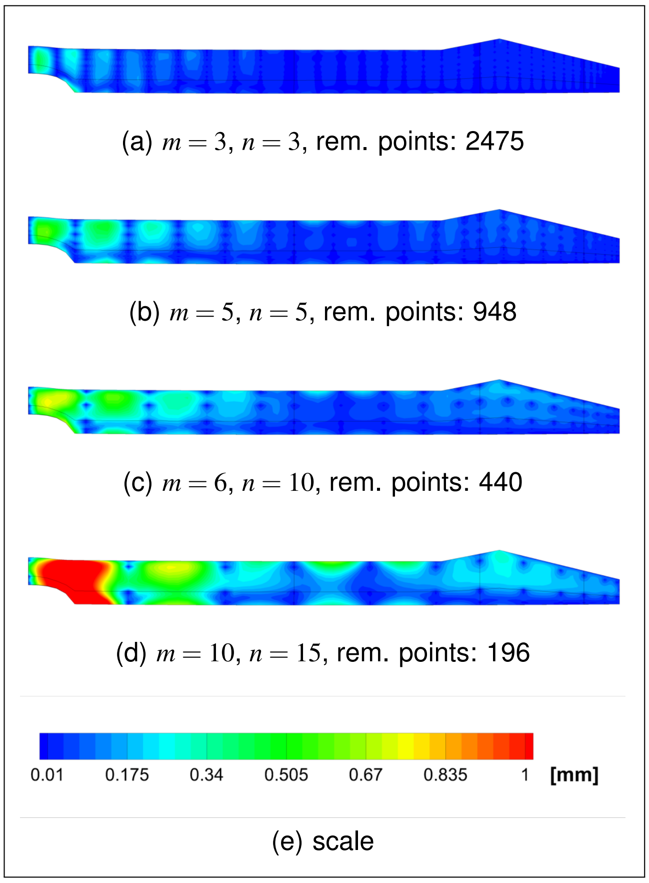
(b) reduced with $m = 6$ and $n = 10$, remaining points: 440

(c) profile section with $n = 10$ (red squares are the remaining points)

Fig. 3. Surface reduction

Figure 4 shows distances of the reconstructed surface compared to the surface directly deformed with the CSD data. It can be seen, that a reduction to $m = 6$ and $n = 10$ still results in an acceptable surface mesh, with deviations below production tolerances, whereas 4(d) would be to coarse for an adequate surface reconstruction. As the discretization gets finer towards the outer edge of the rotor blade, being the area where the main aerodynamic effects take place, a more precise surface deformation is achieved there. On the opposite, a better resolution towards the root would be desirable.

But this shows also, that optimally, the surface node reduction algorithm should select a representative amount of surface points statistically equally distributed on the surface.



(a) $m = 3$, $n = 3$, rem. points: 2475

(b) $m = 5$, $n = 5$, rem. points: 948

(c) $m = 6$, $n = 10$, rem. points: 440

(d) $m = 10$, $n = 15$, rem. points: 196

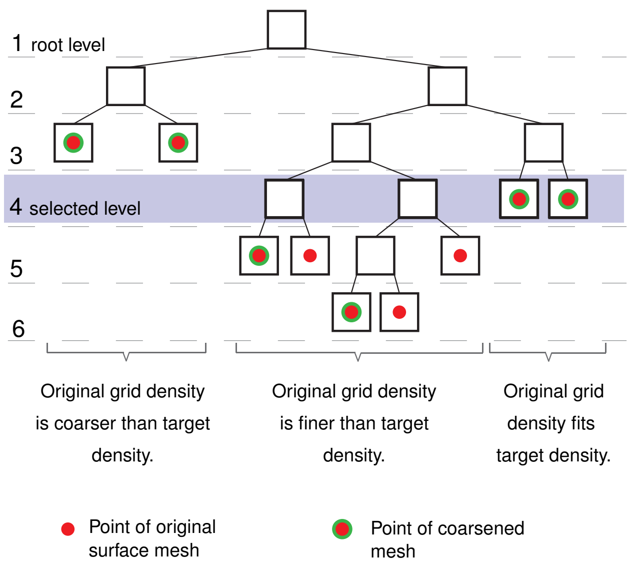
(e) scale

Fig. 4. Qualities of surface reduction using the simple method, showing the offset to deforming all surface points with the CSD data

3.2 Reduction of surface points by using a three dimensional search tree (octree)

For making the method capable to handle unstructured surface meshes and to achieve a regular reference point distribution, an octree algorithm[11] has been implemented.

Sorting all surface points of the original non-deformed surface into a three-dimensional binary tree allows selecting a subset of points by their location in space relative to their neighbors without evaluating



1 root level
2
3
4 selected level
5
6
Original grid density is coarser than target density.
Original grid density is finer than target density.
Original grid density fits target density.
Point of original surface mesh
Point of coarsened mesh

Fig. 5. Exemplary illustration of point selection by means of an one-dimensional search tree

connectivity. For this purpose, preliminarily an empty cube in space is defined, whose dimensions must cover the complete mesh setup. This cube is representing the root of the search tree now being created. All cubes the search tree is built of may contain only one point of the surface or alternatively eight finer cubes. After sorting the first surface point, consequently the search tree consists out of one cube with the initial dimensions containing one surface point. When sorting the second surface point, the root cube is divided into halves for each direction in space. It contains now eight subcubes and no surface point anymore. The first surface point is assigned to the subcube, which covers its position. If the same subcube is covering the location of the first and second surface point, this subcube will be bisected the same way, the root cube was. This procedure will be recursively continued, until both points are not covered by the same cube anymore. Then the third surface point will be sorted the same way.

After all surface points have been sorted, a tree of cubes in space will result, whose branches consist of a different number of levels. The dimensions of the cubes depend on their level inside the octree:

$$l_{edge}(n_{level}) = \frac{l_{rootedge}}{2^{(n_{level}-1)}} \tag{9}$$

Starting with an initial root cube edge length of 100m the second level represents cubes with an edge length of 50m, the third level represents cubes with an edge length of 25m, and so on.

For coarsening the grid, a specific level of the octree can be selected by defining the maximal cube length. In the top example selecting a cube length

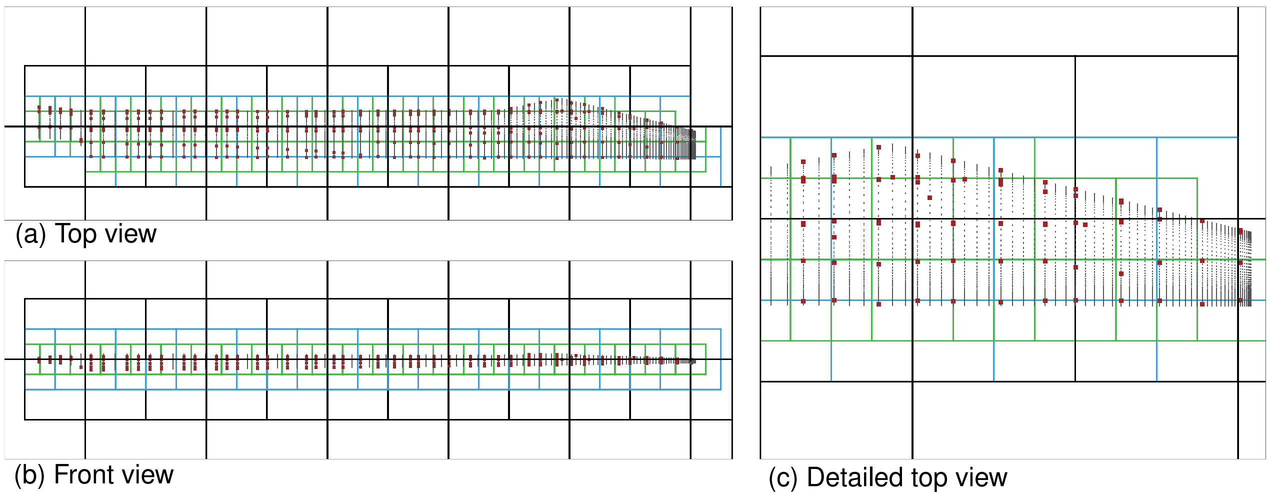


Fig. 6. Original non-reduced surface mesh, resulting coarsened point cloud, and local octree structure

of 0.1m would return all cubes of the 11th level with an edge length of $97.7 \cdot 10^{-3}m$. The coarsened grid will result by selecting the surface point contained by each cube of the selected level. If a branch ends before the selected level, all points covered by this branch are selected. That way, in areas, where the grid node density is coarser than the target density all grid nodes will be selected as reference points. If a cube contains no surface point but subcubes, an arbitrary point of the surface inside its covered space must be selected. All three cases are exemplarily illustrated for a one dimensional binary tree in figure 5.

For convenience, in the current implementation always the first branch is followed recursively down until its end returning one discrete point. In general this will not ensure an equally distributed coarsened point cloud representing the surface, as especially for structured meshes, the sequence the surface points are sorted depends on its i-j-k-location inside the CFD mesh. In other words, it is not ensured, that the averaged distance of the points inside the selected point cloud is approximately equal to the edge length of the selected cube level. However, it is ensured that the maximum distance between two points of the coarsened surface representing point cloud is the double of the edge length of the selected octree's cube level.

The method complexity of the octree sorting and selection algorithm is $(n_{surf} \cdot log(n_{surf}))$. It needs only to be executed when one of the original non-deformed surfaces were modified. In this case, the M_{bb} matrix must be recreated and inverted. Accordingly, the additional computational costs for sorting and selecting are not relevant.

Figure 6 shows an overview and a detail of the original surface mesh of a rotor blade, the resulting coarsened surface point cloud and the borders of

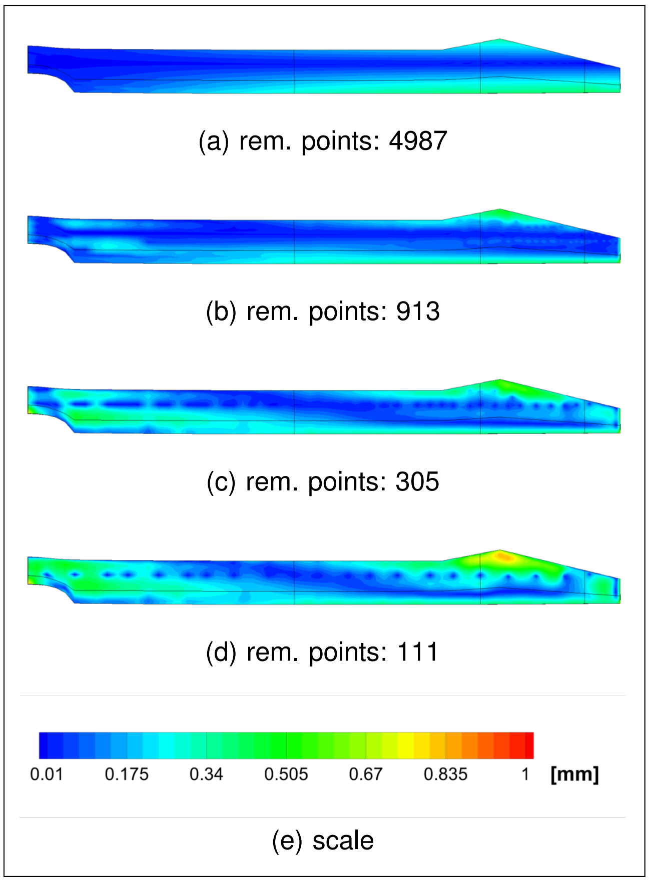


Fig. 7. Different qualities of surface reduction using octree coarsening, showing the offset to deforming all surface points with the CSD data

cubes used for coarsening. The dimensions of the root cube were specified to 100x100x100m. As target density 50mm was specified, which is represented by the 12th level of the octree. This ensures a maximal distance between two points in the resulting point cloud of 100mm.

In comparison to the reduction algorithm based on structured surface patches explained in section 3.1, the accuracy of reconstructed surface is improved, especially near the root section, as shown in figure 7. This allows a further reduction of the number of surface points needed for surface reconstruction and consequently improves the performance.

4 EXAMPLES OF DEFORMATION STRUCTURES

4.1 Overlapping meshes

One major feature of the new deformation algorithm is the ability to handle overlapping meshes. The first example shows a rotor blade that was split into two structures, one for the inner root section and one for the main blade area. The two structures share a certain part of the blade surface, which is needed by the Chimera technique for the transfer between both volume meshes. Figure 8 shows the overlapping area with an exact share of the mesh nodes. Figure 9 has a non-matching surface in terms of mesh nodes; the discretized surface of course is the same within discretization accuracy.

Figure 10 shows a setup of a slotted trailing edge flap of a rotor blade. Four multi-block grids are visible, which are all combined to deform simultaneously. An air gap is represented between the blade and the flap.

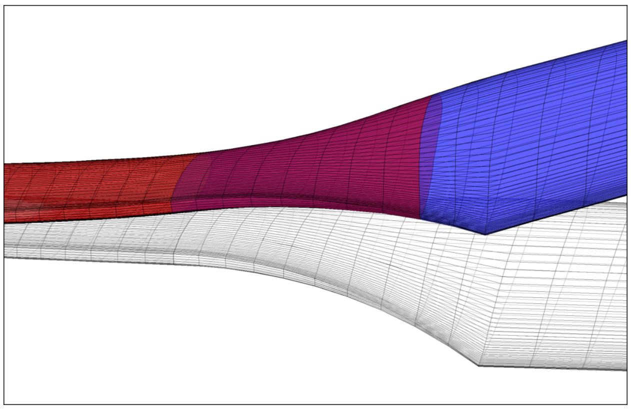


Fig. 8. Matching overlapping surface using Chimera in non-deformed and deformed state

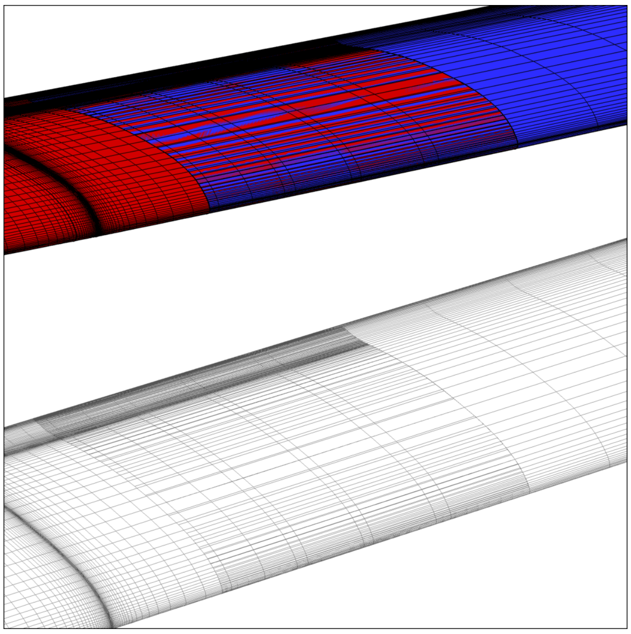


Fig. 9. Non-matching overlapping surface using Chimera in non-deformed and deformed state

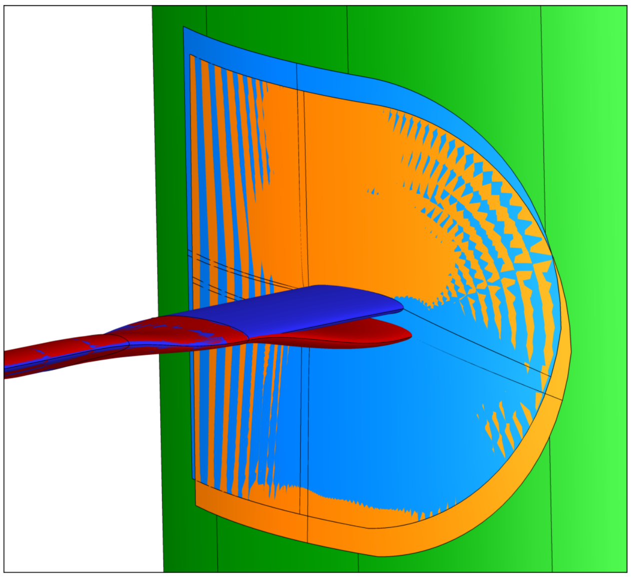


Fig. 11. Rotor-only simulation w/o gap between blade and cylindrical structure (note that the green, light blue and orange surfaces are congruent)
red/orange: old implementation with spline hinge connection
blue: new implentation with shifting of the hinge structure

4.2 Hinges

The new implementation allows to move points along a specifically shaped surface, e.g. a cylinder or a sphere, as demonstrated in figure 11. Note that this represents a Chimera setup where the hinge walls of the rotor blade mesh move along the surface of the background mesh.

Therefore, preliminarily the deformed surface of the rotor blade will be reconstructed from the defor-

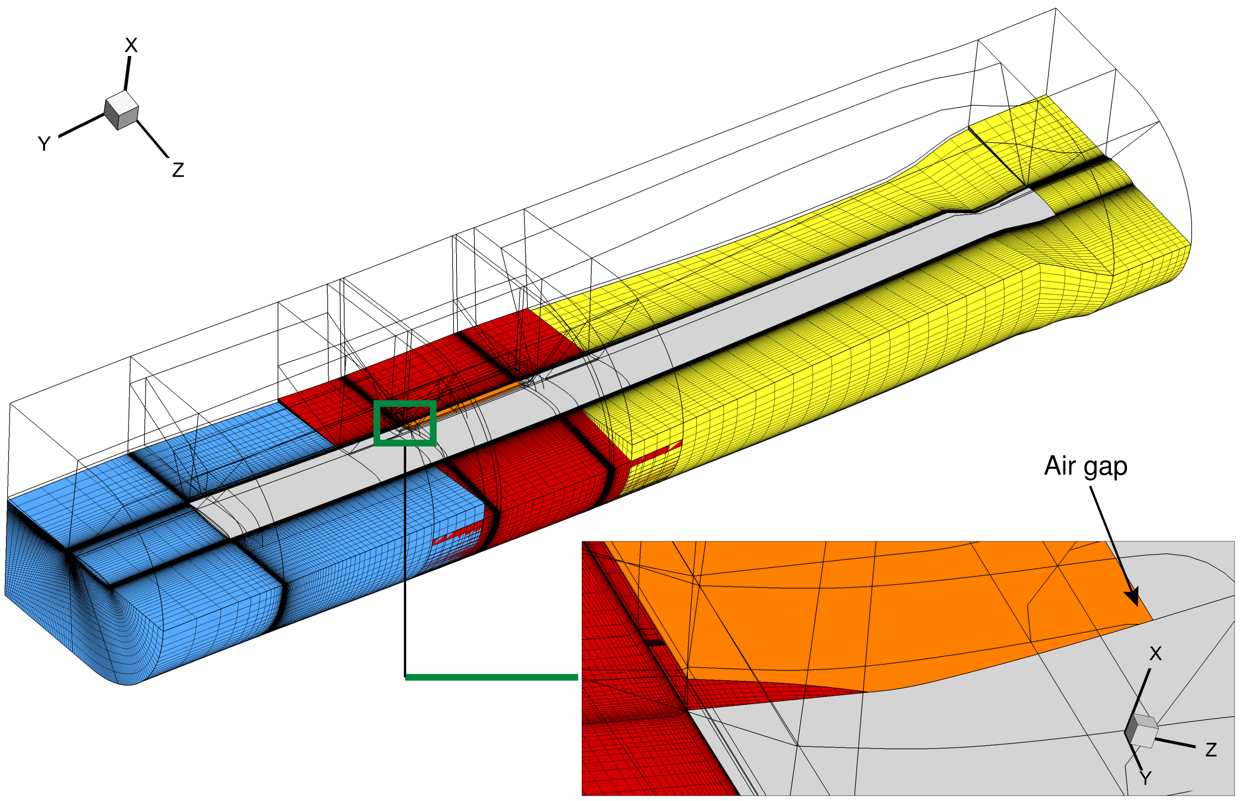


Fig. 10. Slotted trailing edge flap setup using overlapping mesh structures

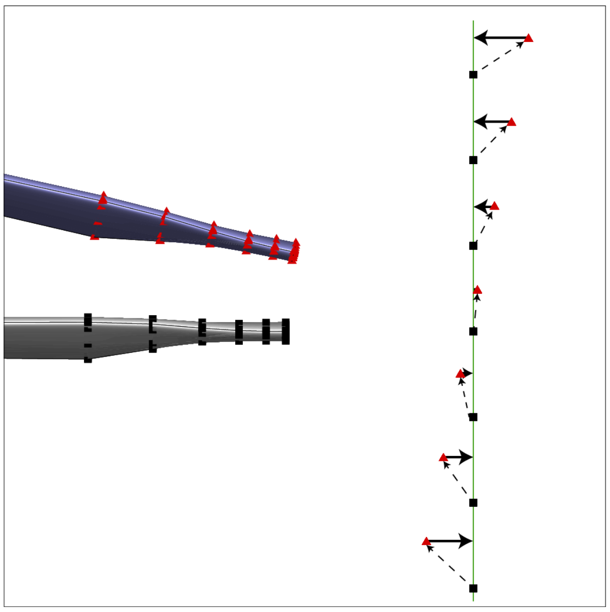


Fig. 12. Reprojection of undeformed (black) points to a virtual surface (green line); red triangles indicate where the deformation would have put them

mation information supplied by the CSD tool. The deformed surface is reduced and the RBF coefficients are regularly computed. Then a set of points along the surface of e.g. a cylinder is spanned and afterwards deformed by applying the recently computed coefficients. The new locations of the points spanned on the cylinder surface are in general not on its envelope surface. Thus the points are reprojected to the surface of the cylinder (figure 12). The originally spanned points are now added to the non-deformed original surface point set of the rotor blade. The final reprojected position of these points is interpreted as their deformed location. Using this extended surface point set, new RBF coefficients are computed replacing the ones used for moving the points spanned on the cylinder. If these newly computed coefficients are applied to a 3D-CFD mesh, the surface points will lie on the deformed surface of e.g. the rotor blade and the points of the CFD mesh originally laid on the sliding surface or next to it will still lie on this surface - just on an other location, which represents the deformation of the originally deformed surface. This procedure is implemented for planes, cylindrical surfaces and spheres. So all kinds of hinges can be represented.

5 VALIDATION

5.1 Comparing deformation between Hermite and RBF interpolation

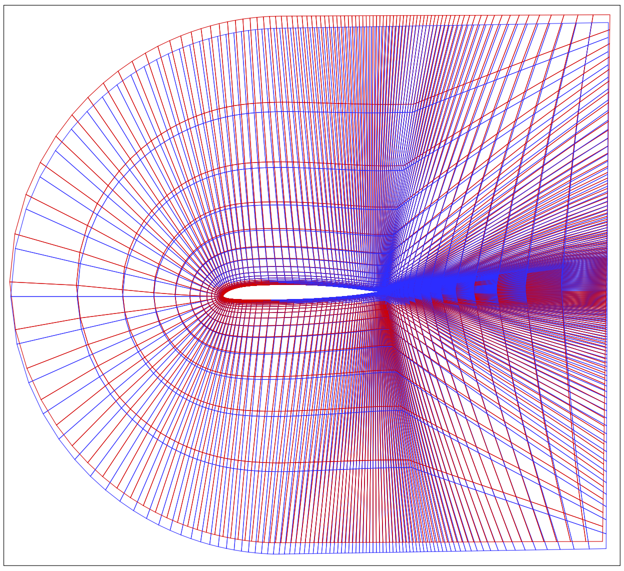


Fig. 13. Difference between Hermite interpolation and RBF interpolation at ca. 0.7 R, red: Hermite interpolation, blue: RBF interpolation

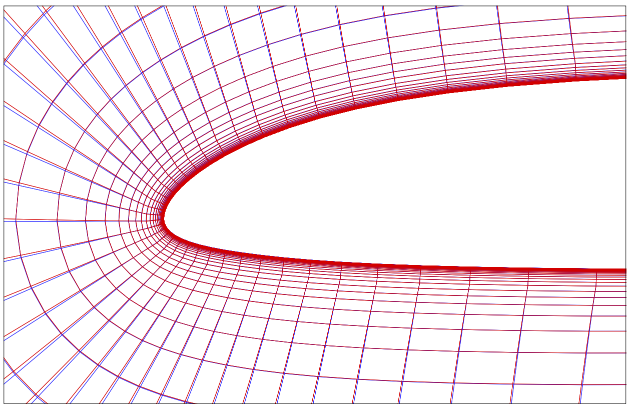


Fig. 14. Difference between Hermite interpolation and RBF interpolation at ca. 0.7 R, closer look at the trailing edge, red: Hermite interpolation, blue: RBF interpolation

The aerodynamic solution shall not be affected by changing the deformation algorithm. In particular, the reconstructed surface must be the same when comparing RBF deformation with the former Hermite Finite interpolation.

Figure 13 and 14 show a slice of the deformed grid. Next to the deformed surface the results of both interpolation algorithms are quite similar. As the Hermite finite interpolation, the RBF Interpolation ensures good grid quality within the boundary layer, as because of its properties the influence of the local deformation is dominant in the area next to the reference points.

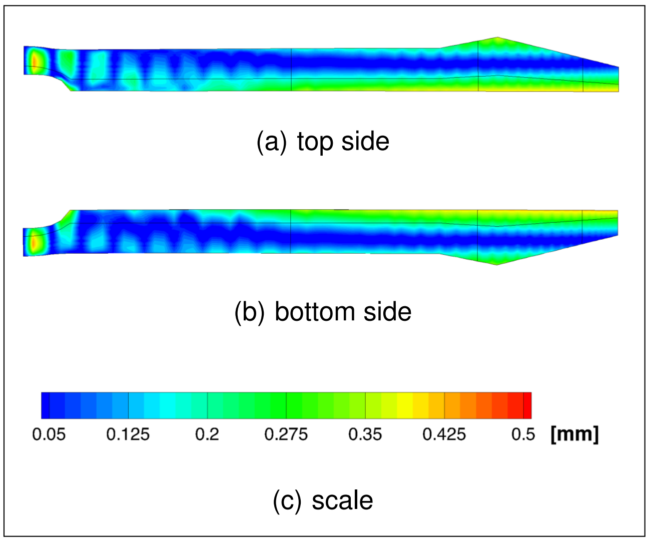


Fig. 15. Difference between Hermite interpolation and RBF interpolation (m=3, n=3), offset plot

Figure 15 shows the difference between a RBF based deformed rotor blade and a rotor blade with the former Hermite interpolation. Clearly visible are the offsets starting from leading and trailing edge and growing towards the outer edge of the blade. These differences come from a misinterpretation of torsion angles for the rigid elements of the Euler-Bernoulli beam within the currently implemented Hermite interpolation algorithm of FLOWer and was found out during validation.

Otherwise, the differences of the deformed surfaces are almost non-existent. The deformation of the volume does not effect its grid quality as shown in figure 13. The differences between Hermite and RBF interpolation are due to a different handling of fixed body motions. The Hermite interpolation algorithm implemented within FLOWer does not include the average torsion angle.

5.2 Trim validation

A trim validation with an isolated five bladed rotor at a flight speed of 125 kts has been conducted. The trim computations were run using an automated script that performs weak-coupling with HOST. As a reference, the former Hermite interpolation deformation was used[4,12,13].

In figure 16 the pitch angles are plotted against the iteration steps for both trim runs. The convergence of these angles is almost the same, with some neglectable discrepancy which is due to a slightly different twist of the elastic rotor blade because of the misinterpretation of the beam angles in the former algorithm (as mentioned above).

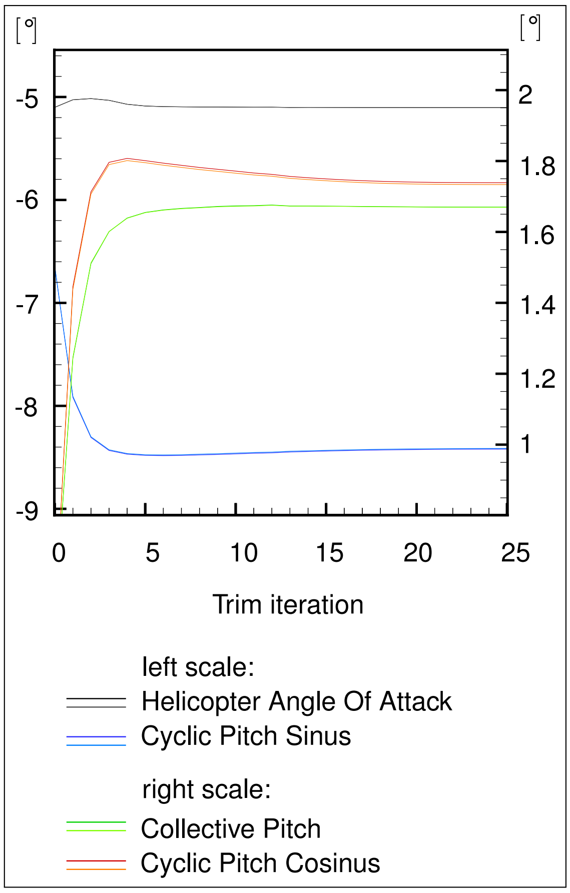


Fig. 16. Angles of pitch during trim iterations (upper color in legend belongs to Hermite interpolation, lower to RBF)

Figure 17 shows the Δc_p at an azimuth of $\varphi = 216°$ between Hermite and RBF algorithm after 24 trim steps which is considered a converged state of the pitch angles (see figure 16). Around $\varphi = 216°$ is one of the few spots, where a slight discrepancy is barely observable, which can be explained by a little shift in the position of flow seperation, leading to different dynamic stall behavior. This is a result of the corrected handling of the elastic torsion angles.

6 CONCLUSIONS

A generalized, high performance mesh deformation library has been developed, that is suitable for structured, unstructured and overlapping meshes. Radial basis functions are used to interpolate displacements of the mesh nodes based on the deformation of a surface. A major achievement is the flexibility, re-usability

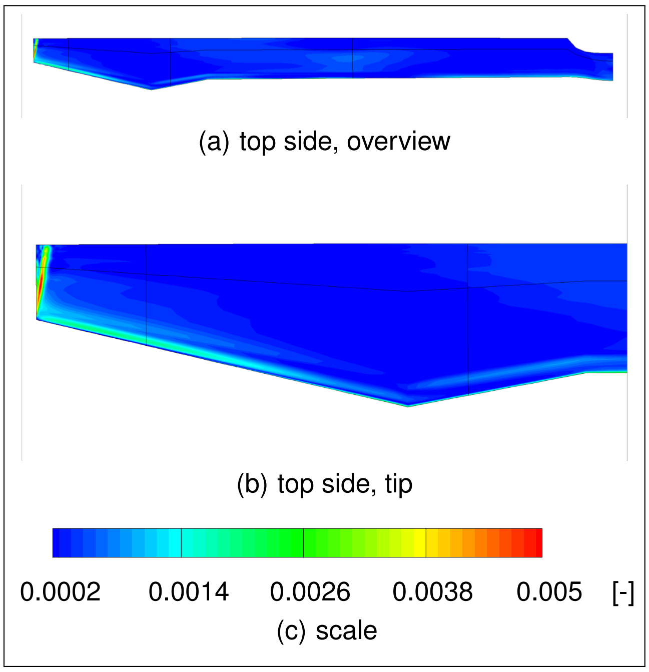


Fig. 17. Δc_p at $\varphi = 216°$ between Hermite and RBF deformation after 24 trim iterations

and the opportunity for use in a highly distributed environment.

The possibility of setting up a slotted flap has been demonstrated and will be investigated in its aerodynamic phenomena in the future. More sophisticated surface setups are thinkable, but exceed the limits of this paper.

The library has been tested on multiple cases of application including trimming a fast-forward isolated rotor flight case in comparison to the previously used and extensively validated algorithm. Results show that the aerodynamic solution is quite the same with a slight difference that come from a misinterpretation in the former application of torsion angles.

Future expansion of the library includes the optimization of the distribution process, so that coefficients are only distributed to the CPUs that need them. A load integration method, that can handle unstructured and overlapping meshes, is currently under development.

Especially when the mesh resolution increases, the importance of a reduction of the surface points becomes a high priority to shorten computation time. The amount of mesh nodes required to perform an adequate mesh deformation only depends on distortion smoothness and spatial extent rather than the spatial discretization. To find a selection of surface points, that have in general the same average distance to each other, an octree based grid coarsening algorithm has been implemented. A fast multipole-method[14] may be applied in the future to reduce the computation time even more.

Further research includes the testing of different basis functions with regard in performance and mesh quality.

Acknowledgments

The authors wish to thank Jonathan Martin for his work on the octree implementation.

REFERENCES

[1] Kroll, N., Eisfeld, B., and Bleeke, H., “The Navier-Stokes code FLOWer,” *Notes on Numerical Fluid Mechanics*, 1999, pp. 58–71.

[2] Benoit, B., Kampa, K., von Grunhagen, W., Basset, P.-M., and Gimonet, B., “HOST, a General Helicopter Simulation Tool for Germany and France,” *Proceedings of the 56th Annual Forum of the American Helicopter Society*, Vol. 56, (2), 2000, pp. 1110–1131.

[3] Johnson, W., *CAMRAD II Comprehensive analytical model of rotorcraft aerodynamics and dynamics*, fourth edition, 2009.

[4] Dietz, M., *Simulation der Umströmung von Hubschrauberkonfigurationen unter Berücksichtigung von Strömungs-Struktur-Kopplung und Trimmung*, Ph.D. thesis, Institut für Aerodynamik und Gasdynamik, Universität Stuttgart, 2009.

[5] Wurst, M., Keßler, M., and Krämer, E., “Detached Eddy Simulation Using the Discontinuous Galerkin Method,” *New Results in Numerical and Experimental Fluid Mechanics IX*, edited by A. Dillmann, G. Heller, E. Krämer, H.-P. Kreplin, W. Nitsche, and U. Rist, Vol. 124, Notes on Numerical Fluid Mechanics and Multidisciplinary Design, Springer International Publishing, 2014, pp. 435–442.

[6] Buhmann, M. D., *Radial Basis Functions: Theory and Implementations*, Cambridge University Press, Cambridge, 8th edition, 2008.

[7] Beckert, A. and Wendland, H., “Multivariate interpolation for fluid-structure-interaction problems using radial basis functions,” *Aerospace Science and Technology*, Vol. 5, (2), 2001, pp. 125–134.

[8] De Boer, A., Van der Schoot, M., and Bijl, H., “Mesh deformation based on radial basis function interpolation,” *Computers & structures*, Vol. 85, (11), 2007, pp. 784–795.

[9] Hounjet, M. and Meijer, J., *Evaluation of elastomechanical and Evaluation of elastomechanical and Evaluation of elastomechanical and Evaluation of elastomechanical and aerodynamic data transfer methods for non-planar configurations in computational aeroelastic analysis*, National Aerospace Laboratory NLR, 1995.

[10] Rendall, T. and Allen, C., “Efficient mesh motion using radial basis functions with data reduction algorithms,” *Journal of Computational Physics*, Vol. 228, (17), 2009, pp. 6231–6249.

[11] Samet, H., “The Quadtree and Related Hierarchical Data Structures,” *ACM Comput. Surv.*, Vol. 16, (2), June 1984, pp. 187–260. doi: 10.1145/356924.356930

[12] Hierholz, K.-H., *Ein numerisches Verfahren zur Simulation der Strömungs-Struktur-Interaktion am Hubschrauberrotor*, Ph.D. thesis, Institut für Aerodynamik und Gasdynamik, Universität Stuttgart, 1999.

[13] Buchtala, B., *Gekoppelte Berechnung der Dynamik und Aerodynamik von Drehflüglern*, Ph.D. thesis, Institut für Aerodynamik und Gasdynamik, Universität Stuttgart, 2002.

[14] Rokhlin, V., “Rapid Solution of Integral Equations of Classic Potential Theory.” *Journal of Computational Physics*, Vol. 60, (2), 1985, pp. 187–207.