MODELING OF DYNAMICALLY MOVING CONTROL SURFACES FOR REALISTIC AIRCRAFT CONFIGURATIONS

L. B. Streher[†*], R. Heinrich[†]

[†] Deutsches Zentrum für Luft- und Raumfahrt, Institute of Aerodynamis and Flow Technology
Lilienthalplatz 7, 38108 Braunschweig, Germany

[*] University of Groningen, Bernoulli Institute for Mathematics, Computer Science and A.I.
Nijenborgh 9, 9747 AG Groningen, The Netherlands

Abstract

The design and optimization of aircraft entail the assessment of multidisciplinary phenomena through high-fidelity multidisciplinary simulations. The realization of such simulations requires the modeling of movable control surfaces for realistic configurations based on two approaches: combining mesh deformation with either overset (Chimera) meshes or sliding interfaces. The generation of Chimera meshes, however, is particularly cumbersome for realistic aircraft configurations, while the formulation of a sliding interface method for cell-vertex finite volume codes is nontrivial. Here we further develop the methodology to automatically generate the overlapping regions required by the overset meshes technique to address aircraft configurations with adjacent control surfaces. Moreover, we develop a sliding interface method inherent to CFD codes with a cell-vertex metric. Both the improved methodology for generating overlapping regions and the sliding interface method are tested for steady and unsteady computations with and without mesh deformation. As good results are obtained for all test cases, important progress is achieved in modeling dynamically moving control surfaces for realistic aircraft configurations.

Keywords

Computational fluid dynamics · Control surface modeling · Chimera technique · Sliding interfaces

1. INTRODUCTION

Computational fluid dynamics (CFD)-based multidisciplinary analysis of aircraft is of major importance for aircraft design and optimization. The execution of aeroelastic simulations, the computation of aircraft trim, and the simulation of high-fidelity flight maneuvers along with the quantification of the related loads are just a few examples of CFD-based multidisciplinary problems in the aeronautical field. The realization of such analyses, however, is still hampered by the difficulties encountered in modeling dynamically moving control surfaces when realistic configurations are to be considered, i.e., when spanwise gaps between lifting and control surfaces are present.

Many approaches have already been proposed to model the dynamic deployment of control surfaces, such as mesh deformation, the combination of overset grids with either rigid body rotation or with mesh deformation, and the combination of sliding interfaces with mesh deformation [5]. Although many approaches are already available, several still fail in some key aspects: mesh deformation is unable to consider the spanwise gaps due to the occurrence of highly sheared grid elements already at low deflection angles, and the combination of overset meshes with rigid body rotation requires the introduction of streamwise gaps between control and lifting surfaces, which are usually sealed in realistic configurations. Combining mesh deformation with either overset meshes or sliding interfaces are currently the most promising approaches. These two methods are, therefore, investigated here.

Although the combination of mesh deformation with overset meshes is a promising approach, generating valid high-quality overset meshes for realistic configurations is still cumbersome and very time-consuming [12]. To address this drawback, we developed a methodology to extend matching patched surfaces, automatically generating the overlapping regions required by the overset grid method (see [14]). This methodology was implemented as an add-on of the finite volume software DLR TAU [13] and resulted in the so-called *AutoLap* software [14]. *AutoLap*, however, was previously unable to generate overlapping regions for configurations with adjacent control surfaces, wherein all boundaries of each control surface grid block (except wall boundaries) were matching patched surfaces. In this work, we further develop *AutoLap* to account for these complex cases. Moreover, we also further develop *AutoLap* in order to increase its efficiency and consequently re-

doi: 10.25967/550049

duce the computational time required to automatically generate the overlapping regions.
In addition to the further development of AutoLap, we also investigate the combination of mesh deformation with sliding interfaces. Sliding interfaces are patched block boundaries present in multi-block grids, which occupy the same spatial footprint [10], but are not necessarily composed of the same surface elements. In order to account for such boundaries, information between the adjacent grid blocks must be transferred. In principle, two methods can be used to allow such communication: the halo interpolation method and the conservative boundary flux interpolation method [7].
Both interpolation methods rely on the creation of halo regions, in which the primitive variables are interpolated based on information from the adjacent grid block. These methods differ essentially in the manner in which the flux is computed on the cell faces located on the sliding boundaries. Whilst the former uses the interpolated primitive variables to compute the fluxes through the boundary faces, the latter uses area-weighted functions of the fluxes from the adjacent grid block to compute the boundary face flux [7]. Although a conservative treatment of sliding boundaries should in general be used to ensure the correct prediction of discontinuous solutions, e.g., in the case of shocks, Fenwick and Allen [7] noted that conserving the fluxes across the sliding boundary has negligible effect on the final solution for steady and unsteady flows. Hence, there is no significant benefit in using the substantially more complex conservative method [7].
Many halo interpolation methods are already available [3, 7, 9, 11], albeit most of them are unsuitable to be implemented in the DLR TAU finite volume solver, as this CFD code uses unstructured grids with a cell-vertex scheme with dual control volumes for the spatial discretization. An appropriate method for such grid metric would be, however, the one developed by Blades and Marcum [3]. Their methodology rely on the creation of a one-cell halo overlapping region composed of prisms (for triangular surface elements) or hexahedrals (for quadrilateral surface elements). The primitive variables on the halo nodes are, then, determined by linear interpolation, whereas the flux is computed based on the interpolated data. Although Blades' and Marcum's sliding interfaces algorithm [3] would be suitable for the DLR TAU code, we prefer to avoid the creation of any halo overlapping region to prevent extrusions of the grid near solid walls, as would occur when simulating realistic aircraft configurations.
As a means of avoiding the creation of overlapping regions, McNaughton et al. [11] developed an algorithm that treats the sliding interfaces as internal Dirichlet boundaries, in which the coupling between the two grid interfaces is achieved using interpolated values of the flow variables on the boundary cell faces. As this method is inherent to cell-centered finite volume solvers, it cannot be used in cell-vertex codes such as the DLR TAU code. For codes in which the flow variables are stored in the grid nodes, however, the sliding interfaces can be intrinsically regarded as a Riemann problem. In this work, we further elaborate on this idea by developing a sliding interfaces algorithm that does not required the creation of overlapping halo regions and is based on the utilization of a Riemann solver to compute the flux through the cell faces located on the sliding boundaries.
In order to validate the sliding interface method and AutoLap's underlying methodology, we compute a Prandtl-Meyer expansion fan, a NACA0012 wing with a generic aileron, and the DLR MULDICON unmanned combat aerial vehicle configuration with inboard and outboard elevons. Finally, the results obtained with both methodologies are compared with each other or with reference results obtained with a single-block grid, and the applicability of these methodologies to realistic aircraft configurations is evaluated.
The outline of this paper is as follows: Section 2 provides a description of the methodology used by AutoLap to automatically generate the overlapping regions required by overset meshes. Next, in Section 3, the sliding interface method for unstructured meshes with a cell-vertex scheme for the spatial discretization is described. Thereafter, in Section 4, both methodologies are tested and validated. Finally, in Section 5, the current work is summarized, and further directions of study are suggested.

2. AUTOLAP: AUTOMATIC OVERLAPPING REGION GENERATOR

AutoLap is a software that aids in the generation of valid high-quality overset grids by automating the generation of the overlapping regions required by the Chimera technique. Although this software was already successfully tested in a previous work (see [14]), it was unable to generate overlapping regions for aircraft configurations with adjacent control surfaces, wherein all boundaries of each control surface grid block (except wall boundaries) were matching interfaces. Here, we further develop AutoLap to take such cases into consideration and, for the sake of completeness, we also describe the entire software anew.
The methodology for the automatic generation of overlapping regions is based on the idea of Blades and Marcum [3]: extruding the faces along a grid block interface into the adjacent grid block based on the connectivity information of the abutting grid block. Here, this idea is only applied for matching patched surfaces, i.e., surfaces that occupy the same spatial footprint and are composed of the same surface elements. Consequently, the overlapping regions can be created by simply adding the elements of the adjacent grid block to the connectivity list of the considered grid block.
As a starting point for AutoLap, a multi-block grid with matching patched surfaces is imported into the software. The number of cell layers to be added to either

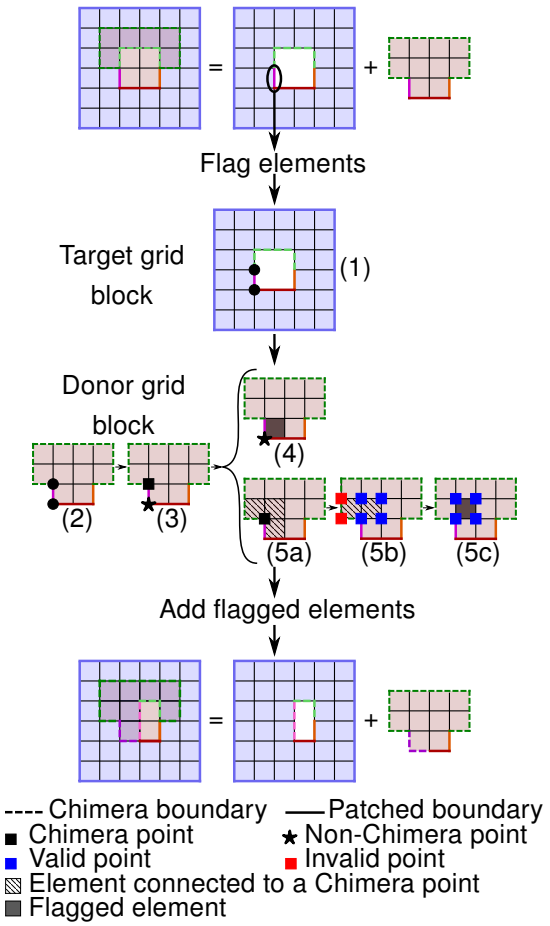


FIG 1. Basic workflow of AutoLap. The input grid consists of two grid blocks with following boundaries: three patched boundary pairs (solid pink, red, and orange lines), two chimera boundaries (dotted dark and light green lines), and one arbitrary boundary (solid blue line).

a specific patched surface or to all patched surfaces is then specified by the user. Next, AutoLap identifies the patched surface pairs (patched surfaces with the same spatial footprint) available on the input grid using bounding boxes (see [14]).

Once all patched surface pairs are identified, the flagging process of the layers of elements to be added to the (target) patched surface can start (see fig. 1). First, the points located on the target patched surface are identified (see step (1) on fig. 1). Next, the corresponding donor points are flagged (see step (2) on fig. 1). These points lie in the same position as the target points, but belong to the donor grid block, i.e., grid block that contains the partner patched surface. The donor points are, then, classified into so-called *Chimera* and *non-Chimera points* (see step (3) on fig. 1). The former are the points that lie on a Chimera boundary (either the donor point lies on a Chimera boundary on the donor grid block or the corresponding target point lies on such a boundary on the target grid block), whereas the latter are the points that do not lie on any Chimera boundary.

Having classified the points as *Chimera* or *non-Chimera*, the volume elements can subsequently be flagged. All volume elements connected to the *non-Chimera* points are directly flagged, as these elements are never present on the target grid block (see step (4) in fig. 1). For the *Chimera points*, the flagging process is carried out carefully in order to avoid the generation of duplicate elements. First, all volume elements connected to the donor *Chimera points* are identified (see step (5a) in fig. 1). In case any of the these elements has already been flagged by a *non-Chimera point*, it is no longer considered in the flagging process (see steps (4), (5a), and (5b) in fig. 1). All remaining elements are further analyzed by classifying their composing grid nodes into so-called *valid* or *invalid* (donor) *points* (see step (5b) in fig. 1).

Valid (donor) *points* are grid nodes that either have no corresponding points on the target grid block (bottom right blue square on step (5b) of fig. 1), or those that have such corresponding points provided that they meet one of the following criteria: (1) they do not lie on a Chimera boundary (top left and right blue squares on step (5b) of fig. 1); (2) they lie on a Chimera boundary, provided that they simultaneously lie on a matching interface (bottom left blue square on step (5b) of fig. 1), and/or their corresponding target point belongs to a Chimera surface (bottom left blue square on step (5b) of fig. 1). The *invalid points*, on the other hand, are those that do not meet the criteria to be considered *valid points*.

If the element that was previously identified using the *Chimera points* consists exclusively of *valid points*, it can be either directly flagged (if all points do not lie on domain boundaries), or further analyzed. This additional analysis is based on the usage of an alternating digital tree (ADT) [4] of the grid nodes on the target grid block and verifies whether a corresponding element exists on the target grid block. If this is not the case, the volume element is flagged (see step (5c) in fig. 1).

Once the volume elements are flagged, the faces that form them are analyzed in order to accommodate the extended boundary. In this work, we introduce the concept of *pseudo patched surfaces* to enable the utilization of AutoLap for cases with adjacent control surfaces. Pseudo patched surfaces are block boundaries that do not necessarily occupy the same footprint as other patched surfaces, although they consist of surface elements identical to those on a (real) patched surface (see dotted lines on the red grid block in fig. 2). Due to the distinct nature of pseudo and real patched surfaces, they are treated differently in the following surface flagging process.

The process of flagging the surface elements is performed for each face that composes the previously flagged donor volume element and differs for faces that abut a volume element and faces that abut a surface element. For the faces that abut a volume element on the donor grid block, AutoLap first verifies whether a corresponding surface element is available on the target grid block. In case such a surface is indeed available, the surface element on the

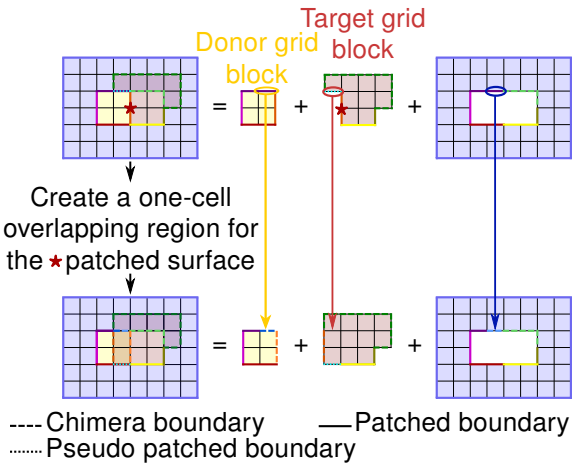


FIG 2. AutoLap: surface element treatment. The input grid consists of three grid blocks with following grid boundaries: six patched boundary pairs (purple, pink, red, orange, yellow, and ocher solid lines), two chimera boundaries (dark and light green dashed lines), and one pseudo patched surface (light blue dotted line).

target block is marked for later deletion, as the underlying boundary is shifted by one cell layer. Otherwise, AutoLap creates a new surface element with a Chimera boundary condition unless the abutting volume element has been previously flagged. For the faces that abut a surface element on the donor grid block, AutoLap first checks whether the surface element lies on a real or on a pseudo patched surface on the target grid block. In case the surface element does not lie on such boundaries on the target grid block, AutoLap creates a new surface element on this grid block, whose boundary condition is specified as either the same as in the donor grid block or as a pseudo patched boundary (if the surface element on the donor grid block lies on a patched boundary - see the pseudo patched surface on the red grid block after creating a one-cell overlapping region in fig. 2). If the surface element lies on a patched surface on the target block, this element is marked for later modification of the boundary type to Chimera, whereas the corresponding patched surface on the target grid block is marked for later deletion (see treated patched surface - indicated with a star - in fig. 2). The same treatment is applied for the surface elements located on the target and donor grid blocks in case the surface element lies on a pseudo patched surface on the target grid block (see yellow and red arrows in fig. 2). In addition to treating these elements, however, the corresponding surface element on the grid block containing the partner surface of the (real) patched surface is also marked to become a Chimera boundary (see blue arrow in fig. 2).

If the specified matching patched surface is to be extended by more than one cell layer, the entire flagging process is repeated until the user-defined number of cell layers is reached. In case the cell layer being flagged is not the first one, the point-based flagging process that forms the basis for the volume element search is, however, based on the surface elements marked on the previously flagged layer [14].

As soon as the flagging process is completed, the flagged elements can be added to the target grid block. First, the points that are not present on the target grid block and that are required to construct the flagged elements are created at the same location as the underlying donor points. Second, volume elements that are topologically the same as the flagged volume elements are created on the target grid block. Next, the surface elements that have been flagged for deletion are removed from the grid connectivity. Subsequently, the surface elements flagged on the last cell layer are created on the donor grid block. Finally, the boundary type of the surface elements that were either flagged for later modification or newly created is updated/set to Chimera. The overlapping region is, then, successfully created for the specified matching interface. If the partner matching interface or other matching interfaces are also to be treated, the same process is to be applied. Otherwise, the grid with the automatically generated overlapping region(s) can be exported and used for CFD simulations [14].

3. SLIDING INTERFACE METHOD

Combining mesh deformation with sliding interfaces is a promising approach to model dynamically moving control surfaces. Implementing a sliding interfaces algorithm in codes that rely on a cell-vertex grid metric (as it is the case of the DLR TAU code [13]) is, however, nontrivial. Here, we propose a method to compute sliding interfaces for multigrid unstructured cell-vertex codes, in which no halo overlapping region is generated.

Figure 3 illustrates a simple structured multi-block grid consisting of two grid blocks wherein the interfaces between the grid blocks are non-matching patched surfaces. Here, each cell-vertex stores two different states of the primitive variables: a CFD state and an interpolated state. The former is intrinsic to the finite volume solver and represents the current value of the primitive variables obtained by the CFD solver, whereas the latter represents the primitive variables computed by interpolations from the CFD solution on the adjacent grid block. Here, two distinct approaches are used to obtain the interpolated state: one for the finest grid of the multigrid method and other for the coarse grids utilized by this acceleration technique.

A first order linear interpolation method is used to obtain the interpolated state for the finest grid. First, the surface quadrilaterals on the block interface, if available, are divided into surface triangles. This division ensures that each surface element lies within a plane and can thus be described by a single normal vector. Please note that this applies only for linear surface elements, i.e., elements where the nodes are connected by lines.

Once the block interface consists only of surface triangles, the interpolation process can begin. To

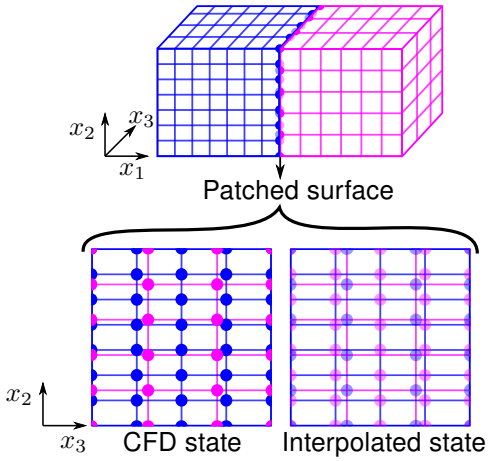


FIG 3. Structured grid composed of two grid blocks (in pink and in blue). The interface between the grid blocks is a non-matching patched surface, in which two states of the primitive variables are available: a CFD state (dark-colored circles) and an interpolated state (light-colored circles).

demonstrate this procedure, we consider the target point t located on the patched surface belonging to the pink grid block (see fig. 3), as depicted in fig. 4.

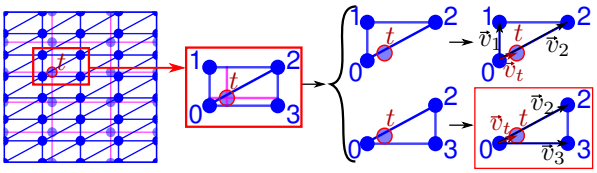


FIG 4. Linear interpolation process for the finest grid.

First, alternating digital trees (ADT) [4] are used to identify the surface triangles that lie on the adjacent patched surface (blue grid block) and surround the target point t. As a certain tolerance is used in this search procedure, the ADT can retrieve more than one surface triangle, as illustrated in fig. 4.
Next, the unit normal vectors of the retrieved surface elements are computed with

$$\vec{n}_{ij} = \frac{\vec{v}_i \times \vec{v}_j}{\|\vec{v}_i \times \vec{v}_j\|}, \tag{1}$$

where $\vec{n}_{ij}$ is the normal vector of the surface element composed of the points 0-i-j, $\vec{v}_i$ is the vector between the point 0 and i, $\vec{v}_j$ is the vector between the point 0 and j, and i and j represent the grid node index of the points that compose the surface triangle.
If the target vector $\vec{v}_t$ (vector between the grid nodes 0 and t) is orthogonal to the normal vector of the surface triangles, i.e., if

$$\vec{n}_{ij} \cdot \vec{v}_t = 0, \tag{2}$$

then the target point lies on the same plane as the surface triangle. The fulfillment of such condition, however, does not necessarily imply that the target point lies within the surface triangle. To verify that, the vectors composing the triangle are linearly combined to obtain the target vector:

$$\vec{v}_t = c_i \vec{v}_i + c_j \vec{v}_j, \tag{3}$$

where c_i and c_j are the scaling coefficients.
The system of linear equations (eq. (3)) is solved using Cramer's rule and the scaling coefficients c_i and c_j are found. If $c_i \in [0,1]$, $c_j \in [0,1]$, and $c_i + c_j \leq 1$, the target point lies within the triangle, which is further referred to as the donor triangle. In fig. 4, the identified donor surface triangle can be distinguished by the surrounding red rectangle.
Once the donor surface triangle is identified, the interpolated state of the primitive variables can be determined by linear interpolations using the computed scaling coefficients. For the velocity in the streamwise direction, for instance, the interpolated state is obtained with

$$\begin{aligned} u_t^{int} &= u_0^{CFD} + c_i(u_i^{CFD} - u_0^{CFD}) \\ &\quad + c_j(u_j^{CFD} - u_0^{CFD}), \end{aligned} \tag{4}$$

where u_t^{int} is the interpolated streamwise velocity on the target point t, and u_0^{CFD}, u_i^{CFD}, and u_j^{CFD} are the streamwise velocities computed by the CFD solver at the grid nodes 0, i, and j, respectively. The interpolated states of the remaining primitive variables are obtained in a similar manner.
Unlike the computation of the interpolated state for the finest grid, this state is obtained in a much simpler manner for the coarse grids. First, k-d trees [2] are used to identify the nearest neighbor grid node (on the adjacent grid block) of the target point under consideration. Then, zero-order interpolations are performed based on the values of the primitive variables at the previously identified nearest neighbor grid node. Thereby, the interpolated state of the primitive variables on the target point is obtained.
As the DLR TAU code [13] is a parallelized solver that operates with distributed memory, the donor surface triangle (for the finest grid) and the nearest neighbor grid node (for the coarse grids) are not necessarily located on the same rank as the target point. Therefore, information about the sliding interfaces must be exchanged between ranks. To avoid communication overheads, distinct Message Passing Interface (MPI) communicators are created for each patched surface pair, i.e., grid block boundaries that occupy the same spatial footprint. These MPI communicators consist only of the ranks that contain information about the patched surface pair under consideration, thus avoiding unnecessary communication with ranks that do not store any information about the considered patched surface pair.
Upon the computation of the interpolated states for each grid point located on the block interfaces, the first part of the sliding interfaces problem is solved. As the CFD and interpolated states are now directly available, the sliding boundaries can be seen as a Riemann problem, in which the CFD and interpolated states are equivalent to the right and left states of the

Riemann problem. Hence, the inviscid fluxes through the boundaries can be computed with flux-vector splitting or flux-difference splitting schemes. Here, these fluxes are computed with the modified AUSM method proposed by Kroll and Radespiel [8], due to the improved capability of this discretization scheme to predict viscous flows accurately.
In this work, the viscous fluxes through the sliding interfaces are not considered as the contribution of such fluxes is usually negligible compared to the inviscid fluxes in regions with small velocity gradients.

4. VERIFICATION TESTS FOR AUTOLAP AND THE SLIDING INTERFACE METHOD

In order to test the new developments of AutoLap and the sliding interface method, three distinct test cases are analyzed here. First, steady and unsteady Prandtl-Meyer expansion fans are investigated. The unsteady computation is particularly relevant as it replicates the relative body motion that occurs during the deployment of a control surface through the sliding motion in the spanwise direction of the interface boundary between the grid blocks. Next, a NACA0012 wing with a generic aileron and spanwise gaps between the lifting and control surfaces is examined. The aileron is deployed by $\delta = -5°$ and $\delta = -20°$ to ensure that the methods developed here also work for moving control surfaces. Finally, the DLR MULDICON unmanned combat aerial vehicle (UCAV) configuration with inboard and outboard elevons is considered. This test case is crucial to validate the new developments of AutoLap and its usability to generate overlapping regions for configurations with adjacent control surfaces. Moreover, it also demonstrates the applicability of the sliding interface method for realistic aircraft configurations. All computations are carried out using the DLR's finite volume code TAU [13], which uses a cell-vertex grid metric and provides an already consolidated implementation of the Chimera method.

4.1. Prandtl-Meyer expansion fan

The inviscid supersonic flow of air about a concave corner is investigated here. This flow is gradually deflected to follow the contour of the concave wall, generating a set of Mach waves and consequently a Prandt-Meyer expansion fan.
The Euler equations are discretized in space with a second-order accurate finite volume method, in which the convection term is discretized with a central scheme with added matrix dissipation. The air flow is characterized by the following free-stream properties: pressure of $p_\infty = 101325\,\text{Pa}$, temperature of $T_\infty = 273.15\,\text{K}$, and Mach number of $\text{Ma}_\infty = 1.5$. The two-dimensional expansion fan geometry is characterized by a turn angle of $\theta = 7.6°$, an inlet height of $H_{in} = 2.3\,\text{m}$, an outlet height of $H_{out} = 2.5\,\text{m}$, and a width of $L_x = 2\,\text{m}$, whereby the expansion corner lies at $0.25L_x$.

Two sets of three grids each are analyzed here: one set of structured grids (see fig. 5) and one of unstructured grids. For each set of grids, three grids are analyzed: a *reference* single-block grid, a *patched* dual-block grid in which the sliding interface method is applied on the block interface, and a *Chimera* dual-block grid in which the overlapping regions were automatically generated by AutoLap. Here, only the structured grids are illustrated for the sake of brevity.

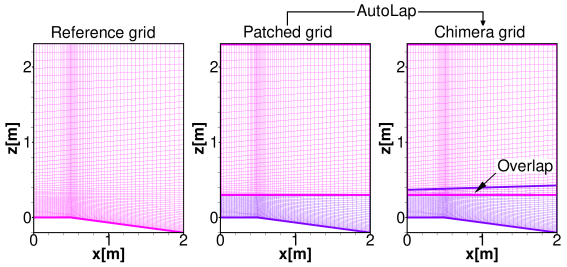
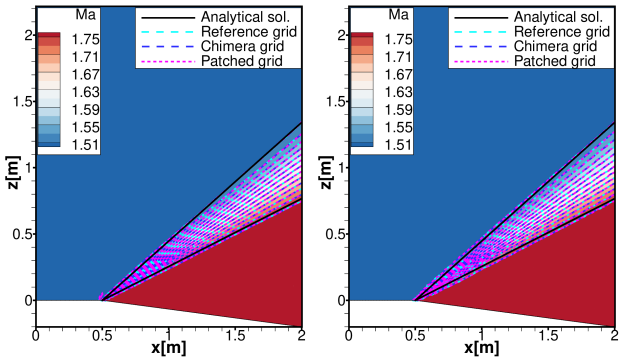


FIG 5. Structured grids for the Prandtl-Meyer expansion fan test case. The overlapping regions of the Chimera grid are created with AutoLap, whereby the patched grid is used as input.

In addition to the CFD computations using the previously described grids, the location of the forward and rearward Mach lines is also calculated analytically according to the inviscid compressible theory. This analytical solution is used here for validation purposes.

4.1.1. Steady-state

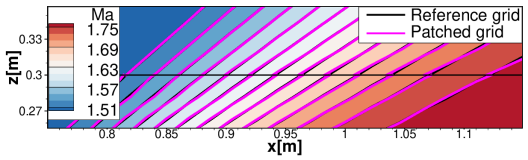
The Mach number distribution obtained for the steady-state simulations of the expansion fan with both sets of grids is illustrated in fig. 6 along with the analytical solution.



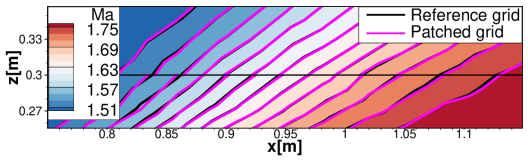
(a) Structured grids. **(b) Unstructured grids.**

FIG 6. Mach number distribution of the steady-state expansion fan.

The results obtained with the reference, patched, and Chimera grids agree very well for both structured and unstructured grids. The location of the forward and rearward Mach lines is also reproduced relatively well when comparing to the analytical solution. The main difference between the computed solutions appears when using the sliding interfaces algorithm, as a small discontinuity is present on the block interface, as illustrated in fig. 7. This discontinuity, however, is to

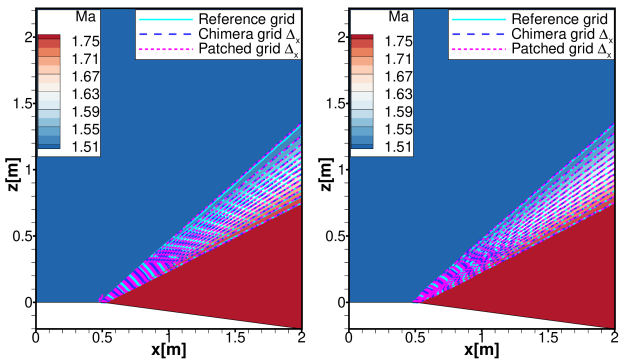


(a) Structured grids.



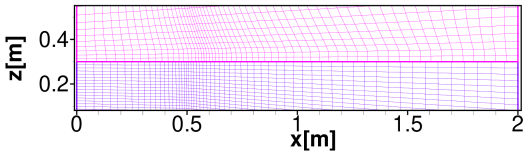
(b) Unstructured grids.

FIG 7. Mach number distribution at the block interface for the reference and patched grids.

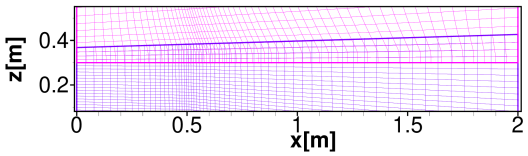


(a) Structured grids. (b) Unstructured grids.

FIG 8. Mach number distribution of the expansion fan with a sliding boundary for the non-deformed reference grid and the Chimera and patched grids deformed by $\Delta x = 0.1\,\mathrm{m}$.



(a) Patched structured grid.



(b) Chimera structured grid.

FIG 9. Interface between the top and bottom grid blocks after a mesh deformation of $\Delta x = 0.1\,\mathrm{m}$.

be expected since the sliding interface method used here is based on the solution of a Riemann problem involving distinct states on each side of the boundary.

4.1.2. Sliding boundary

To assess the suitability of AutoLap and the sliding interface method to cases involving relative body motion, here the interface/Chimera block boundary of the upper grid block (depicted in pink in fig. 5) slides in the streamwise direction according to a sinusoidal function with a maximal amplitude of $\Delta x = 0.1\,\mathrm{m}$. The sliding motion is generated by the deformation of the grid according to a thin plate spline radial basis function [6]. The unsteady flow marches in time using an implicit second-order dual time-stepping approach, in which the pseudo time steps are solved with an explicit Runge-Kutta scheme.

The Mach number distribution for the non-deformed reference grids, as well as the patched and Chimera grids deformed by $\Delta x = 0.1\,\mathrm{m}$, is illustrated in fig. 8. Due to the deformation of the meshes, the patched boundaries no longer match and the grid elements on the overlapping region are no longer identical for the upper and lower grid blocks (see fig. 9). Although the non-deformed reference grids and the deformed patched and Chimera grids are not directly comparable anymore, a good agreement is observed between the solutions obtained with these grids. Hence, both AutoLap and the sliding interface method are suitable for use in cases involving relative body motion, such as the deployment of control surfaces.

4.2. NACA0012 wing with a generic aileron

Here, AutoLap and the sliding interface method are analyzed using a finite NACA0012 wing with a generic aileron. The wing has a chord length of $c = 1\,\mathrm{m}$ and a spanwise length of $L_w = 3.05\,\mathrm{m}$, whereby the generic aileron is located at $0.69 L_w$ and has a spanwise length of $L_{ail} = 0.8\,\mathrm{m}$. Gaps with a spanwise length of $L_{gap} = 0.01\,\mathrm{m}$ are present between the lifting and control surfaces.

The flow around the wing is governed by the Navier-Stokes equations, which are numerically approximated using a finite volume method. First- and second-order accurate upwind spatial discretizations of the convective term are used to initialize the simulations. Once a reasonable physical solution is obtained, this method is switched to a second-order accurate central scheme with added matrix dissipation. The turbulence is modeled according to the one-equation Spalart-Allmaras turbulence model in its negative form [1].

All computations are performed with an angle of attack of $\alpha = 5°$ and the air flow is characterized by the following free-stream properties: Reynolds number of $Re_\infty = 7 \times 10^6$, Prandtl number of $Pr_\infty = 0.72$, and Mach number of $Ma_\infty = 0.7$. For the deployment of the control surface, a thin plate spline radial basis function is used [6].

Hybrid grids consisting of four grid blocks and a hole geometry are utilized in this work, as illustrated in fig. 10. From these fours grid blocks, one is unstructured (plain wing grid block - in black in fig. 10) and three are structured (grid blocks for the aileron, and inboard and outboard spanwise gaps - in pink, blue, and purple in fig. 10, respectively). The structured grid blocks are used here to provide a good resolution of the vortices caused by the presence of the spanwise gaps, whereas the hole geometry is introduced on the

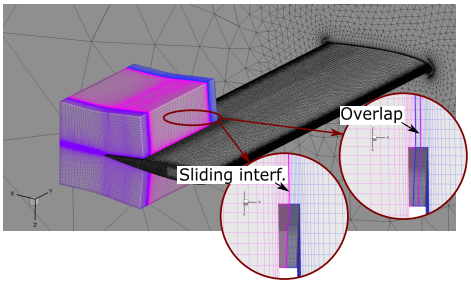


FIG 10. Hybrid grids for a NACA0012 wing with a generic aileron. Detailed views depict the grid with sliding interfaces (on the left) and the Chimera grid (on the right).

surface of the wing where the aileron and the gaps are located to incorporate these features to the plain wing.
A patched and a Chimera grid are used for the CFD computations (see fig. 10). The former applies the sliding interface method to the block interfaces between the gap and the aileron grid blocks, whereas the latter was created with AutoLap using the patched grid as input grid. Here, the results obtained with the Chimera grid serve to validate those using the sliding interface method, as AutoLap has been validated for this test case by Streher and Heinrich [14] and the modifications of AutoLap do not influence the overlapping region generation for this NACA0012 wing.

4.2.1. Aileron deployed by $\delta = -5°$

The CFD solution of the NACA0012 wing with the aileron deployed by an angle of $\delta = -5°$ is computed for both patched and Chimera grids. The obtained pressure coefficient and eddy viscosity distributions are illustrated in fig. 11, whereas fig. 12 shows the pressure coefficient curves for four wing cross sections, including two cross sections at the location of the patched boundary (cross sections at y_1 and y_3). Very good agreement is found between the results obtained with the patched and Chimera grids.

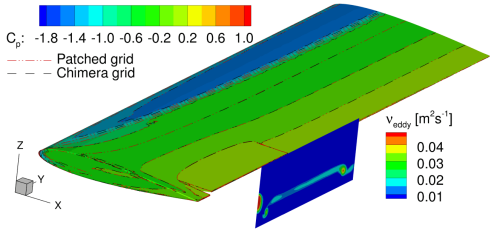


FIG 11. NACA0012 wing with a generic aileron deployed by $\delta = -5°$: pressure coefficient C_p and eddy viscosity ν_{eddy} distributions for the patched grid, and pressure coefficient isolines for the patched and Chimera grids.

4.2.2. Aileron deployed by $\delta = -20°$

To further test the sliding interface method, the generic aileron is deployed by an angle of $\delta = -20°$ and the obtained results are illustrated in figs. 13 and 14. Here, we also observe a really good agreement between the solutions computed with the Chimera and patched grids. These results indicate that the sliding interfaces algorithm is indeed suitable to account for the sliding movement of block interfaces even for viscous flows.

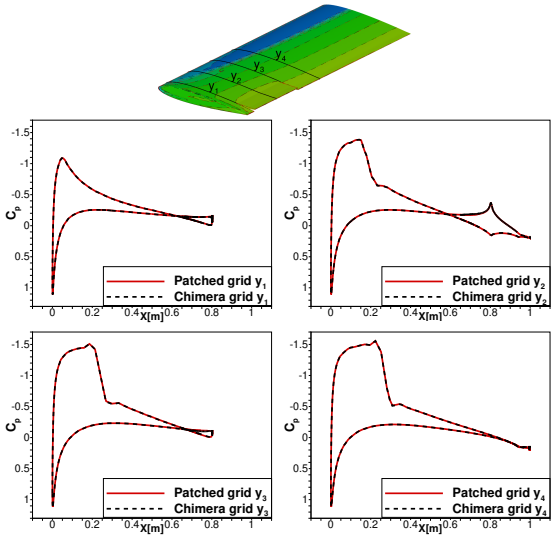


FIG 12. NACA0012 wing with a generic aileron deployed by $\delta = -5°$: pressure coefficient curves for four wing cross sections of the patched and Chimera grids.

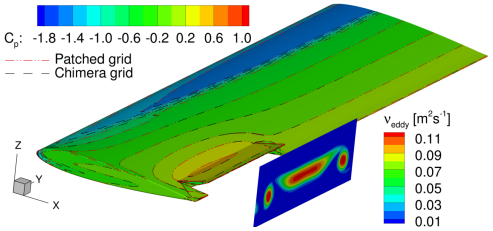


FIG 13. NACA0012 wing with a generic aileron deployed by $\delta = -20°$: pressure coefficient C_p and eddy viscosity ν_{eddy} distributions for the patched grid, and pressure coefficient isolines for the patched and Chimera grids.

4.3. DLR MULDICON UCAV half configuration with inboard and outboard elevons

The DLR MULDICON unmanned combat aerial vehicle configuration with inboard and outboard elevons is considered here aiming at testing especially the modification of AutoLap to account for adjacent control surfaces. For that purpose, three grids are generated: a single-block reference grid (see fig. 15a), a patched grid (see fig. 15b), and a Chimera grid (see fig. 15c). The patched grid employs the sliding interface method on the patched block boundaries, whereas the overlapping regions of the Chimera grid were automatically generated by AutoLap in about nine minutes using the patched grid as input.
The Navier-Stokes equations are solved with a finite volume method in which the convective terms are

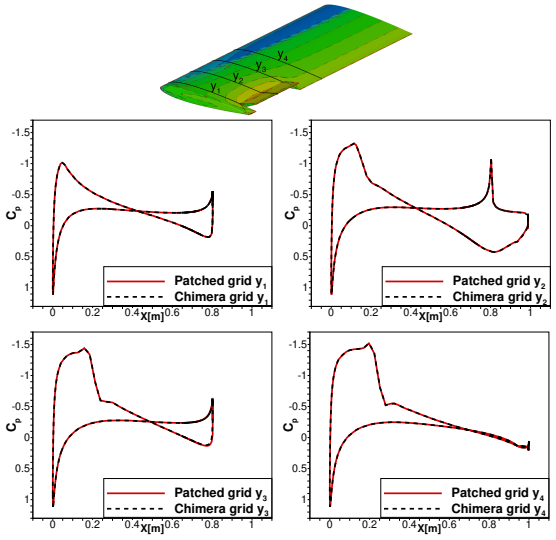
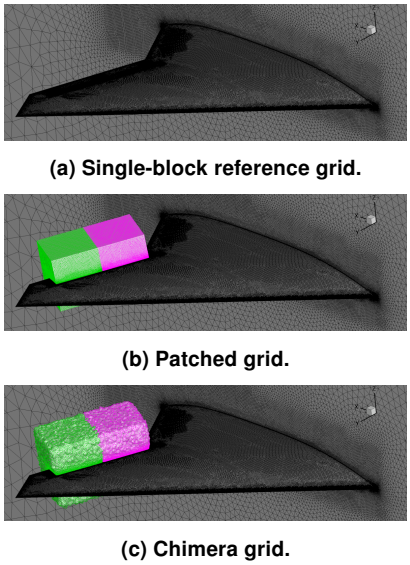


FIG 14. NACA0012 wing with a generic aileron deployed by $\delta = -20°$: pressure coefficient curves for four wing cross sections of the patched and Chimera grids.



(a) Single-block reference grid.

(b) Patched grid.

(c) Chimera grid.

FIG 15. Unstructured grids for the MULDICON configuration with adjacent elevons. The patched and Chimera grids are composed of three grid blocks: one for the UCAV configuration (in black), one for the inboard elevon (in pink), and one for the outboard elevon (in green).

discretized using a second-order accurate central scheme with added matrix dissipation and the turbulence is modeled with the Spalart-Almarras model in its negative form [1]. All computations are performed with an angle of attack of $\alpha = 2°$ and the following free-stream properties: Reynolds number of $Re_\infty = 55.8 \times 10^6$, Prandtl number of $Pr_\infty = 0.72$, and Mach number of $Ma_\infty = 0.4$.
The obtained pressure coefficient distribution for the MULDICON configuration is illustrated in fig. 16. A very good agreement between the results obtained for all tested grids can be observed. This indicates that the modified AutoLap is well suited for such complex configurations. Furthermore, it proves the suitability of the sliding interface method for realistic aircraft configurations.

5. CONCLUSIONS

This work set out to facilitate the modeling of dynamically moving control surfaces when using the combination of mesh deformation with either overset (Chimera) meshes or sliding interfaces for realistic aircraft configurations, i.e., configurations that consider the spanwise gaps between lifting and control surfaces. To accomplish this, we have pursued two strategies: the further development of the methodology to automatically generate the overlapping regions required by the Chimera technique and the development of a sliding interface method for CFD codes that employ a cell-vertex grid metric.
The methodology used by the automatic overlapping region generator software (AutoLap) has been further developed to allow its application to aircraft configurations with adjacent control surfaces. To this end, the concept of pseudo patched surfaces has been introduced in AutoLap. These surfaces are treated differently than (real) patched surfaces as they do not necessarily occupy the same spatial footprint of another patched surface located in a different grid block. The use of pseudo patched surfaces thus overcomes difficulties when the creation of an overlapping region results in the generation of surface elements that are adjacent to a patched surface on the donor grid block, as would be the case for configurations with adjacent control surfaces. In addition to enhancing AutoLap with respect to abutting control surface blocks, we have also introduced alternating digital trees to optimize the performed searching procedures.
To use the combination of mesh deformation with sliding interfaces, a new sliding interface method for cell-vertex finite volume codes has been developed and implemented in the DLR TAU code [13]. This method is based on the evaluation of two distinct states of the primary variables at the same grid node: a CFD state and an interpolated state. The CFD state is computed directly by the finite volume solver, while the interpolated state is computed through interpolations based on the CFD solution of the abutting surface. As the DLR TAU code [13] uses the multigrid acceleration technique, two distinct interpolation methods have been employed: a linear interpolation for the finest grid and a zero-order nearest neighbor interpolation for the coarse grids. With the availability of the CFD and interpolated states on all grid nodes that lie on a sliding boundary, a Riemann problem is solved to account for the convective fluxes through the sliding interface. For that, we have used the modified AUSM method proposed by Kroll and Radespiel [8].
Both the modified AutoLap and the sliding interface method have been investigated using three test cases. First, steady and unsteady Prandtl-Meyer expansion fans have been studied. Second, a finite

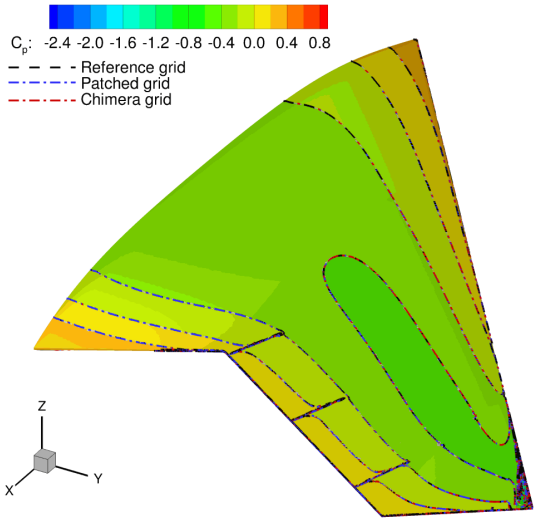


FIG 16. MULDICON configuration: pressure coefficient C_p distribution for the reference grid and pressure coefficient isolines for the reference, patched and Chimera grids.

NACA0012 wing with a generic aileron deployed by $\delta = -5°$ and $\delta = -20°$ has been examined. Finally, the DLR MULDICON UCAV configuration with adjacent elevons has been tested. All these tests yielded very good results, thus validating the methodologies developed here. Therefore, the simulation of moving control surfaces for realistic aircraft configurations has been considerably facilitated.
A natural progression of this work would be to develop a method for computing the viscous fluxes at the sliding boundaries using the CFD and interpolated states already available at each boundary element. In addition, the realization of multidisciplinary analyses using the methods developed in this work should be pursued.

Contact address:

larissa.streher@dlr.de

References

[1] S. R. Allmaras and F. T. Johnson. Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model. In *Seventh international conference on computational fluid dynamics (ICCFD7)*, volume 1902, 2012.

[2] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[3] E. L. Blades and D. L. Marcum. A sliding interface method for unsteady unstructured flow simulations. *International Journal for Numerical Methods in Fluids*, 53(3):507–529, 2007.

[4] J. Bonet and J. Peraire. An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems. *International Journal for Numerical Methods in Engineering*, 31(1):1–17, 1991.

[5] L. A. Capsada and R. Heinrich. Development of the DLR TAU code for modeling of control surfaces. In *Deutscher Luft- und Raumfahrtkongress 2018*, 2019.

[6] J. Duchon. Splines minimizing rotation-invariant semi-norms in Sobolev spaces. In W. Schempp and K. Zeller, editors, *Constructive Theory of Functions of Several Variables*, pages 85–100, Oberwolfach, 1977. Springer-Verlag.

[7] C. L. Fenwick and C. B. Allen. Development and validation of sliding and non-matching grid technology for control surface representation. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 220(4):299–315, 2006.

[8] N. Kroll and R. Radespiel. An improved flux vector split discretization scheme for viscous flows. *DLR-Forschungsbericht*, 93(53):1–47, 1993.

[9] I. MacPherson, J. Rodgers, C. Allen, and C. Fenwick. Sliding and non-matching grid methods for helicopter simulations. In *44th AIAA Aerospace Sciences Meeting and Exhibit*, page 1070, 2006.

[10] D. R. McDaniel, R. H. Nichols, and J. B. Klepper. Unstructured sliding interface boundaries in Kestrel. In *54th AIAA Aerospace Sciences Meeting*, page 1299, 2016.

[11] J. McNaughton, I. Afgan, D. D. Apsley, S. Rolfo, T. Stallard, and P. K. Stansby. A simple sliding-mesh interface procedure and its application to the CFD simulation of a tidal-stream turbine. *International Journal for Numerical Methods in Fluids*, 74(4):250–269, 2014.

[12] L. Reimer and R. Heinrich. Modeling of movable control surfaces and atmospheric effects. In N. Kroll, R. Radespiel, J. W. Burg, and K. Sørensen, editors, *Computational Flight Testing*, volume 123, pages 183–206. Springer-Verlag, 2013.

[13] D. Schwamborn, T. Gerhold, and R. Heinrich. The DLR TAU-code: recent applications in research and industry. In *ECCOMAS CFD 2006: Proceedings of the European Conference on Computational Fluid Dynamics*, 2006.

[14] L. B. Streher and R. Heinrich. Meshing strategy for movable control surfaces: Towards high-fidelity flight maneuver simulations. In A. Dillmann, G. Heller, E. Krämer, and C. Wagner, editors, *New Results in Numerical and Experimental Fluid Mechanics XIII*, volume 151, pages 677–686. Springer International Publishing, Cham, 2021.