JOURNAL OF AIRCRAFT
Vol. 47, No. 1, January–February 2010

Patch Assembly: An Automated Overlapping Grid Assembly Strategy

Florian Blanc[*]
Airbus Industries, 31060 Toulouse, France

DOI: 10.2514/1.44116

This paper presents a strategy called patch assembly for assembling overlapping grid systems. This strategy has been designed to deal efficiently with specific grid assembly cases. These are cases where the chimera technique is used to add a new feature, described in a patch grid, to a geometry defined in a background grid. The patch assembly algorithm automatically performs the blanking of the background grid using simple and robust algorithms. In particular, the algorithm is not based on any geometric mask, which can be difficult to build. The blanking performed ensures a minimum overlap between grids. This grid assembly strategy is applied to a simple two-dimensional case and to a big three-dimensional wing-body configuration for modeling control surfaces. On both configurations, results are compared to results obtained on grid setups manually set with masks. A favorable influence of the minimum overlap produced has been observed on these configurations.

I. Introduction

SOME aerodynamic problems, such as the aerodynamics of control surfaces of civil transport aircraft, require computational fluid dynamics (CFD) on complex geometries: the transonic turbulent flows that are encountered require the use of efficient CFD codes and all the details of the geometry must be represented to obtain accurate results. Moreover, control surfaces are moving parts and any satisfactory numerical method for studying this problem has to be able to manage these changes of geometry.

The chimera technique is a good solution for dealing with such problems. This technique permits different mesh blocks to overlap each other [1], making it easier to generate meshes for complex geometries, such as aircraft with control surfaces, by meshing the different parts of the body separately. The different grids generated can be moved or deformed independently to represent the changes of each geometry. Another big advantage of the chimera technique, and perhaps the most important one, is that it is a capability that can be added to complex CFD codes designed for coincident meshes without large-scale modifications that could reduce their efficiency.

Another important feature of the chimera technique is the possibility to cut holes into meshes, that is, disable the computation of the flow in some cells of a mesh. This feature, called blanking, can be used to remove a part of a given mesh and replace it with a new elementary mesh that overlaps the first one. This is commonly done for adding blades to a helicopter [2,3] or releasable stores under a jet fighter [4]. This can also be done for adding control surfaces such as spoilers and ailerons on a civil transport aircraft wing.

The global chimera preprocessing, during which holes are cut and interpolated regions are selected, is called grid assembly. To make the most of the chimera technique in an industrial context, the grid assembly process has to be automated. In the following, an entirely automatic process, called patch assembly, for performing the grid assembly of an overlapping grid system is described and tested.

Every chimera grid assembly process can be divided into two main operations: the blanking step and the interpolation step.

During the blanking step, the holes are cut and blanked cells are identified. This is generally performed using geometrical entities prescribed by the user, called masks, which define geometrical regions in which cells will be blanked. In some grid assembly algorithms, hole cutting is done automatically.

The interpolation step begins by locating all the cells that have to be interpolated. Even if this location can vary from one grid assembly process to another, some interpolated cells are mandatory. First, cells near the borders of overlapping domains on which no boundary condition is applied must be interpolated (they are usually called fringe cells). Second, cells near the blanked region also have to be interpolated because these regions define some new borders of the computational domain on which boundary conditions have to be set by interpolation [1].

The second part of the interpolation step is to find donor cells for the cells that have to be interpolated. Algorithms that can be described as "geometric preconditioners" are generally used for rapidly locating donor cells. When these algorithms are used, an approximation of the mesh is stored with a logical structure. This approximation is used for the rapid location of candidate cells that can be used for interpolation. These can be an alternating digital tree (ADT) [5], an octree data structure [6], or any other technique. A simpler, but time-consuming alternative to these preconditioned donor search algorithms is the stencil walk algorithm [7], which locates donor cells by iteratively testing cells in the donor mesh until a valid donor cell is found.

The basic grid assembly strategy requires the user to set masks [2] and to use only the mandatory interpolated points previously described. Many solutions are available for implementing these masks. One solution is to define masks using a surface taken from the geometry to blank the cells inside the bodies [8]. The x-ray [9] mask and the direct cut approach [10] are examples of this technique. Another solution is to use a geometric approximation of the body surface [11] for cutting holes. In the Cartesian hole map mask [12], the surface is approximated on a Cartesian grid. Generally, these techniques are efficient but complex to implement and to use, at least because of the thresholds that must be defined for correctly handling a geometry defined by multiple surfaces.

The main issue with mask-based grid assembly strategies is that, when masks are not correctly set, no donor cells can be found for some mandatory interpolated cells, making the computation impossible. This problem can lead the user to define the mask through a trial and error process that can dramatically increase the time required for performing a CFD study on overlapping meshes.

Some alternatives to this basic technique have been created for automating the grid assembly process. Another advantage of these techniques is that they can automatically take into account constraints for improving the quality of the grid setup produced by the grid assembly. Three main families of automated grid assembly algorithms can be distinguished: advancing front algorithms,

Received 3 March 2009; revision received 28 May 2009; accepted for publication 3 August 2009. Copyright © 2009 by F. Blanc. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 0021-8669/10 and $10.00 in correspondence with the CCC.

[*]Ph.D. Student, Aerodynamics Department; florian.blanc@airbus.com.

algorithms based on a cell quality criterion, and algorithms independent of the geometry.

Advancing front algorithms find the blanked cells through an iterative process which moves the borders of the blanked region to ensure a minimum overlap between the grids. The movement of these borders is done to place the overlap region as far as possible from the bodies. This ensures that meshes that are near bodies, which are supposed to be better adapted for computing the flow around these bodies, will be used for the calculation. The minimum overlap produced allows a better convergence rate of the computation [13] and simplifies the integration of forces and moments acting on the body. However, these techniques still require the use of masks for blanking the cells that are inside bodies. The “cut paste” algorithm [14,15] and Wey’s advancing front technique [16] are examples of this type of algorithm.

The next algorithm group includes algorithms based on a cell quality criterion. The principle of these algorithms is to select interpolated cells using a cell quality criterion. For example, in the “implicit hole cutting” (IHC) algorithm [17–19], the selection of interpolated cells is based on the volume of cells: the IHC algorithm generates grid setups in which the finest cells available are used for computation and the coarse cells are interpolated. The main advantage of this algorithm is that it ensures that the finer mesh will be used to enable best flow resolution. Another advantage is that it is very simple to implement because it only requires a donor search algorithm: no blanking method needs to be implemented. One drawback of this algorithm is that the interpolated cells can be numerous, implying high computational costs in terms of time and memory. Another difficult point with this algorithm arises from cells inside bodies which are not blanked by the cell selection process. A mask is required for blanking these cells. Cai et al. [20,21] use a grid assembly strategy similar to the IHC, based on grid density for selecting the interpolated cells, with a mask defined using the body surfaces for removing the cells inside bodies.

It is interesting to note that the criterion used in the advancing front algorithms (criterion based on wall distance) and the criterion used in the IHC for selecting interpolated cells (criterion based on cell volume) can be modified. For instance, a criterion based on cell angle and stretching is used in PEGASUS 5 [22,23] grid assembly software for providing better grid setups.

Algorithms independent of the geometry do not use any geometrical operation for the grid assembly. The grid assembly process is completely driven by the test results concerning the location of the boundary cells and the cells that can be interpolated. The only grid assembly process known by the author that is entirely based on this technique is the one implemented in the CMPGRD [24] (or in Overture in a more recent version) and in Xcog [25]. This software seems to be efficient. However, because of its costly iterative structure and the numerous operations requiring data coming from several elementary grids, it seems that it cannot be easily used for assembling big industrial meshes with several millions of cells on distributed memory parallel computers that are now commonly used in the industry.

The patch assembly algorithm proposed in this paper is a new automated grid assembly algorithm. This algorithm does not aim at being better in general than the other automated mesh assembly algorithms: this algorithm has just been designed to be completely automated and as efficient as possible on a particular type of overlapping grids (described further in this paper). The patch assembly algorithm enters the family of the algorithms independent of the geometry and it has an original structure that makes it adapted for the assembly of big meshes on distributed memory parallel computers. The algorithm is presented and applied only for the assembly of structured chimera grids. Chimera on unstructured meshes is not considered here (for chimera on unstructured meshes, see, for example, Noack and Boger [26] and Togashi et al. [27]).

Section II presents the patch assembly algorithm, followed in Sec. III by a description and analysis of results obtained on grid setups generated with this algorithm.

II. Patch Assembly Algorithm

A. Basis and Principle of the Algorithm

Two types of chimera grids can be distinguished:

1) Multiple group overlapping grids: These are overlapping grids made of numerous elementary grids that overlap each other and where no hierarchy can be established between the elementary grids. In such overlapping grids, an elementary grid has no meaning alone because it meshes only a part of the surface of the geometry. A simple example could be an airfoil mesh where the leading edge, the trailing edge, the upper wing, and the lower wing would be meshed by different overlapping grids. See the work of Slotnick et al. [28] on a space shuttle mesh for a more complex example. This type of grid is encountered mainly when the volume mesh is generated from on an overset surface grid decomposition [29].

2) Patch overlapping grids: In these grids, two subgroups of grids can be distinguished. The first one is called the background grid: it is a complete coincident mesh around a given geometry on which a new feature is to be added. The second subgroup, called the patch grid, meshes a new feature added to the geometry defined in the background grid. The background grid does not necessarily need to be a complete mesh. The only requirement is that it has to be complete in regions that are not covered by the patch. This description covers a wide range of cases. Stores under a military plane, control surfaces of an airplane (see Fig. 1), rotor blades, and ice on an airfoil are some examples of configurations that can be represented using a patch grid added onto a background grid.

The patch assembly algorithm is designed for the second type of overlapping grid. On these grids, the patch assembly algorithm gives priority to the patch grid. This means that the flow will be computed on the patch grid in regions of overlap, while the background grid will be either blanked or interpolated. The hierarchy introduced between the elementary grids is justified by the fact that the patch grid is most of the time better adapted than the background grid for describing the flow near the new features added.

The patch assembly algorithm is only capable of dealing with chimera grids containing one patch grid. That is the reason why it cannot deal with multiple group overlapping grids. As such grids are generally made of more than two subgroups of chimera grids, several patch grids must be considered if a “background grid plus patch

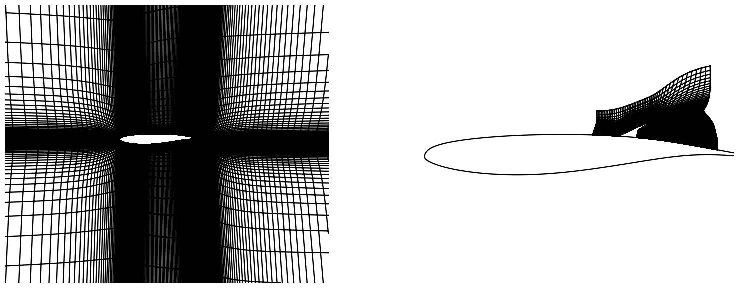

a) Background grid: OAT15A air foil b) Patch grid: a fully deployed spoiler

Fig. 1 Examples of background grid and patch grid.

grids" paradigm is used for assembling these grids. Making the patch assembly capable of dealing with more than one patch grid would be a first step toward the use of this algorithm on various types of chimera grids.

For a pair of background and patch grids, the main necessary steps proposed for completing the grid assembly are as follows:

1) Cells near borders of patch grids with no prescribed boundary conditions have to be interpolated. This allows the flow information to be transferred from the background grid to the patch grid.

2) A blanking has to be performed in the background grid. Around blanked cells, some cells will be interpolated. This will allow the flow information to be transferred from the patch grid to the background grid. As already mentioned, it can be interesting to minimize overlap between grids. The size of the blanked region has to be maximized to fulfill this constraint.

The patch assembly algorithm performs these two steps using the following strategy: Blank as many cells of the background grid as possible, ensuring that the fringe cells around the blanked cells can be interpolated.

To solve the previous problem, the patch assembly algorithm uses three main steps:

1) Perform the donor search for fringe cells on the borders of the patch grid.

2) Find all interpolatable cells of the background grid.

3) Blank as many cells of the background grid as possible:

a) Blank all interior cells of the background grid.

b) Blank all unnecessary interpolatable cells.

Each of these steps will now be described. During this description, it will be supposed that a donor search algorithm is available. The purpose of this algorithm is to find, for a given cell in a mesh, a cell in another mesh that can be used for its interpolation. To ease the description of the patch assembly algorithm, illustrations are restricted to a two-dimensional representative test case. This is a background mesh containing an OAT-15A airfoil [30] on which a spoiler is added using a patch grid (see Fig. 1). For the sake of clarity, only the case of a grid setup adapted to a flow solver based on a five-point centered spatial derivation stencil will be considered. However, the patch assembly algorithm can be easily adapted to any flow solver spatial stencil.

B. Interpolation of Borders of the Patch Grid

Most of the time, borders of the patch grid do not have any defined physical boundary conditions. Cells near these borders need to be interpolated to define boundary conditions. The number of layers of cells that need to be interpolated near these boundaries depends on the size of the spatial derivation stencil. The temporal derivation stencil can be taken into account for moving body cases [31] but this possibility has not been considered here. With the five-point centered stencil considered here, two layers of cells need to be interpolated on these borders. Fringe cells on the border of the spoiler grid are shown in Fig. 2.

Note that the interpolation of these cells is mandatory. As a consequence, the global algorithm can fail if some of these cells are orphan (that is to say if they cannot be interpolated). This situation can mainly happen if one border of the patch grid that must be interpolated enters in a body defined in the background grid (see Fig. 3). For moving body cases, special care must be taken to make sure that this does not happen during the computation while meshes are moved.

Some solutions can be proposed to remove these orphan cells.

First, the surface of the bodies defined in the background grid can be used to perform the blanking (for instance using an x-ray mask) of all cells of the patch grid that are inside these bodies. Such a modification would improve the robustness of the patch assembly algorithm but would reduce its generality. For example, with this modification, the algorithm could not deal with cases where the patch grid is used for adding a cavity inside the body defined in the background grid.

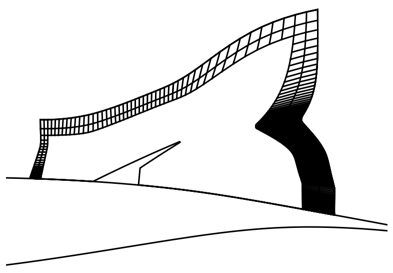
A second solution is used here to remove these orphan cells. The principle of this solution, based on a method proposed by Chesshire and Henshaw [24] and Nakahashi et al. [32], is to iteratively blank all the fringe cells that cannot be interpolated and move the interpolated cells toward the outside of the bodies. The implemented algorithm is the following:

```
1) $\mathcal{I}$ is the list of cells that have to be interpolated
2) $\mathcal{N}i$ is the list of cells that cannot be interpolated
3) $d(A)$ is the list of cells that are in the derivation stencil of cell $A$
4) $isFinished \leftarrow True$
5) **for all** cell $A \in \mathcal{I}$ **do**
6)       interpolate $A$
7)       **if** $A$ cannot be interpolated **then**
8)          add $A$ to $\mathcal{N}i$
9)          $isFinished \leftarrow False$
10)      **end if**
11) **end for**
12) **for all** cell $A \in \mathcal{N}i$ **do**
13)      blank cell $A$
14)      **for all** cell $B$ in $d(A)$ **do**
15)         **if** $B$ is not interpolated nor blanked **then**
16)            add $B$ to $\mathcal{I}$
17)         **end if**
18)      **end for**
19) **end for**
20) **if** $isFinished = False$ **then**
21)      go to 4
22) **end if**
```

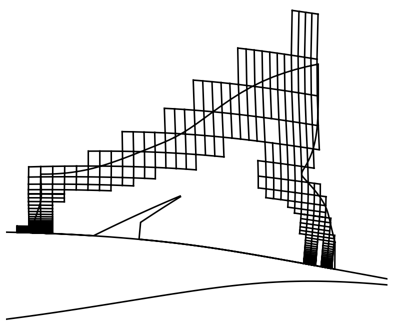
Figure 3 presents the result given by this algorithm on a case where the patch grid is a simple Cartesian grid added on a background airfoil mesh. Some fringe cells of the patch grid are obviously not interpolatable if no correction is applied.

C. Finding All Interpolatable Cells of the Background Grid

A donor search is performed on all cells of the background grid to identify all interpolatable cells. The computational time of this step can be significantly reduced by limiting the donor search to the cells



a) Interpolated cells on the borders of the patch grid



b) Donor cells for the borders of the patch grid

Fig. 2 Interpolated and donor cells for the borders of the patch grid.

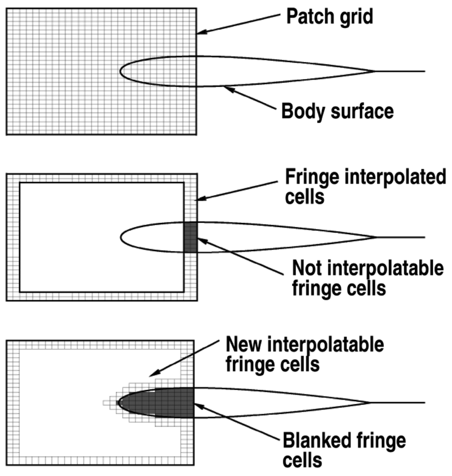

Patch grid
Body surface
Fringe interpolated cells
Not interpolatable fringe cells
New interpolatable fringe cells
Blanked fringe cells

Fig. 3 Correction of the fringe of the patch grid. Top, the patch grid. Middle, fringe cells of the patch grid that are not interpolatable. Bottom, blanked cells and fringe cells after the execution of the algorithm.

of the background grid that are included in the Cartesian envelope of the patch grid. The donor cells of the background grid that are used for interpolating the cells of the fringe of the patch grid are excluded from this search (see Fig. 4). Their interpolation would lead to a wrong grid setup. Isolated cells that can be seen in Fig. 4a are cells that are located between the interpolation stencils of patch grid fringe (visible in Fig. 2b). As these cells do not enter in these stencils, they are candidates for interpolation.

D. Blanking of Cells of the Background Grid

The next step of the patch assembly algorithm is to blank a maximum of cells of the background grids. Note that, according to some authors, this blanking is unnecessary [17]. For example, the IHC algorithm does not produce any blanked cells. Here, the blanking is performed for four main reasons:

1) The computation of the completely nonphysical flow in the unblanked cells inside the bodies can cause the failure of the flow computation.

2) Removing unnecessary interpolated cells reduces the computational cost of the interpolation process, in terms of time and memory.

3) In Lee and Baeder [17], cells inside bodies are not blanked because they are considered as sufficiently far from any other computed cell. Thus, the solution in these interior cells has no influence on the computed cells, if the Courant–Friedrichs–Lewy (CFL) number is sufficiently low. Here, the patch assembly algorithm is associated with a solver that uses an implicit time advance algorithm which allows very high values of the CFL number.

4) Blanking of these cells greatly simplifies postprocessing on the grid setup. Two types of cells have to be blanked can be identified. These are 1) noninterpolatable cells, which are mainly cells lying inside bodies defined by the patch grid, and 2) interpolatable cells that are not necessary to the computation.

Two specific algorithms, described in the next two subsections, are sequentially used for blanking each type of cell.

1. *Blanking of Cells Inside Bodies*

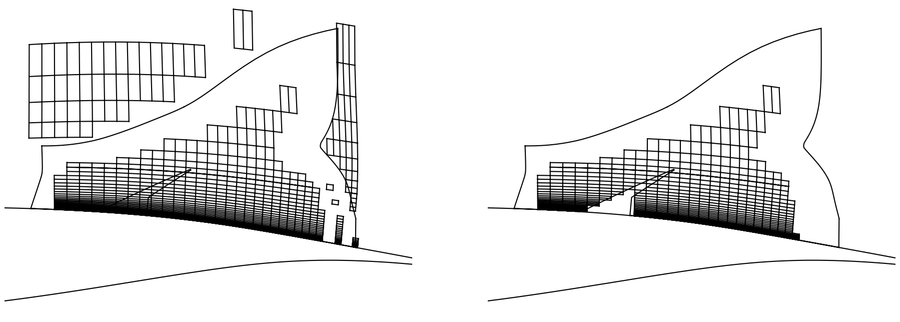
The blanking of cells that are lying inside bodies is a general problem in algorithms for automated grid assembly. The surface of bodies is generally used for defining a mask: for example, an x-ray mask or a "Cartesian hole map" can be chosen. Using these masks, which have to be defined by the user, would reduce the level of automation of the patch assembly algorithm. Thus, an automated algorithm has been developed to solve this issue.

This algorithm is based on the fact, visible in Fig. 4b, that interior cells of the background grid are a group of cells that are completely surrounded by interpolatable cells. The algorithm used is as follows:

```
1) $\mathcal{O}$ is the list of interior cells
2) for all cell $A$ of the background grid that is in the Cartesian envelope of the patch grid do
3)     add $A$ to $\mathcal{O}$
4) end for
5) $isFinished \leftarrow True$
6) for all cell $C$ in the background grid such that $C$ is not in $\mathcal{O}$ do
7)     for all cell $S \in d(C)$ do
8)         if $S \in \mathcal{O}$ then
9)             remove $S$ from $\mathcal{O}$
10)            $isFinished \leftarrow False$
11)        end if
12)    end for
13) end for
14) if $isFinished = False$ then
15)     go to 5
16) end if
17) blank all cells in $\mathcal{O}$
```

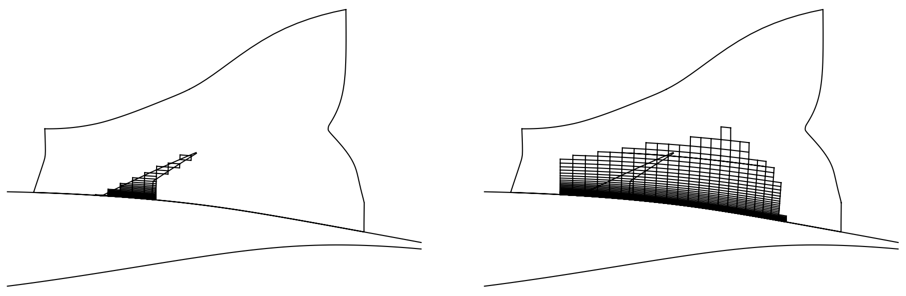
This algorithm can be understood as an "advancing front technique" where cells are removed from the list of interior cells by propagating, toward the region of interpolated cells, the border between the computed cells of the background grid and the interior cells (as defined in step 2 of the algorithm). Interior cells found by this algorithm are presented in Fig. 5a, starting from the state of Fig. 4b.

The main possible case of failure of this algorithm would be a chimera grid where the interior cells of the background grid are not completely surrounded with interpolatable cells. This could occur on the spoiler test case used here, if there was a gap between the airfoil skin defined by the background grid and the airfoil skin defined in the patch grid. In this gap, a layer of noninterpolatable cells would appear in the background grid. This layer would cause the failure of the interior cell detection algorithm by "connecting" interior cells inside the spoiler to the noninterpolatable cells that are far from the patch. To solve this problem near multiple defined skins, a correction of cell coordinates [31,33] can be used.



a) Cells of the background grid that are candidates for interpolation **b) Cells of the background grid that are interpolatable**

Fig. 4 Finding the interpolatable cells of the background grid.



a) **Blanked interior cells of the background grid**

b) **All blanked cells of the background grid**

Fig. 5 Blanking of cells of the background grid.

2. Blanking of Nonnecessary Interpolated Cells

When the overlap between the patch and the background grid is large, there can be a lot of interpolated cells in the background grid. In most of the cases, there is no need to interpolate all these cells. The algorithm described next blanks some of these interpolatable cells.

Generally speaking, when a region of a mesh is blanked, a fringe of interpolated cells is built around the masked cells to apply a boundary condition on the new boundaries created around the mask. The size of this fringe depends on the size of the spatial derivation stencil. With the five-point stencil considered here, the fringe is two points wide. This ensures that no blanked cells will be used in the spatial derivation stencil of any computed cell of the background grid.

The current algorithm blanks as many interpolated cells as possible, keeping the two-point wide fringe of interpolated cells around blanked cells. Note that these fringe cells will be, *by construction*, interpolatable because they are selected among all the interpolatable cells of the background grid. These fringe cells cannot be orphan cells.

The criterion used for finding the unnecessary interpolated cells is as follows: Blank all the interpolated cells of the background grid that do not enter in the spatial derivation stencil of any of the computed cells.

The blanked cells added by this algorithm can be seen in Fig. 5b. The set of blanked cells that appears in this figure ensures a minimum overlap between the chimera grids. The fringe of interpolated cells around the blanked region of the background grid appears by removing the blanked cells from the interpolatable cells (Fig. 4b).

III. Numerical Results

The patch assembly algorithm has been implemented in the elsA [34] flow solver, which already includes some advanced chimera functionalities [3] but no automated overlapping grid assembly algorithm. In particular, the ADT-based donor cell search algorithm was already available for the implementation of the patch assembly algorithm.

The outline of this section is as follows: first, the patch assembly is tested on a simple 2-D case. On this case, results obtained with the patch assembly algorithm are compared with results obtained with a grid setup without overlap minimization (manually done with masks) and with a coincident mesh on both steady and unsteady computations. Lacking any experimental data for this case, the results on the coincident mesh will be considered as a reference in these comparison. This is not too strong of an assumption, as elsA flow solver with coincident mesh has been validated on numerous steady and unsteady simulations [35]. Second, the patch assembly algorithm is applied to a complete aircraft configuration where the algorithm's efficiency is analyzed.

A. NLR7301 Airfoil Under a Wall

1. Steady Flow Computation

The patch assembly algorithm has been tested on a simple two-dimensional case. This case consists of a two-dimensional NLR7301 airfoil under a wind-tunnel wall.

For the overset meshes, two grids were used: one for the wall and the other for the airfoil. See Fig. 6 for the grid setup without overlap minimization obtained using masks, and Fig. 7 for the grid setup produced by the patch assembly algorithm.

For the patch assembly grid setup, the wall mesh was declared as the background grid and the airfoil mesh as the patch grid. For the grid setup without overlap minimization, the mask used was built using the airfoil skin. The large overlap between meshes in this grid setup can be clearly seen. At a first glance, the two chimera grid setups look very different, however, the only real difference is in the amount of blanked cells in the wall mesh. The blanking of the airfoil grid is almost the same for the two grid setups.

Computations were performed on both grid setups. The Reynolds-averaged Navier–Stokes equations with a Spalart–Allmaras [36] turbulence model were solved for a Mach number equal to 0.85, using a Jameson scheme [37] with a lower–upper symmetric successive overrelaxation implicit phase [38]. A multigrid technique [39,40] with one coarse grid was used to accelerate the convergence. The wind-tunnel wall was treated as an Euler wall (slipping wall boundary condition). The airfoil skin was modeled by a Navier–Stokes wall boundary condition with a mesh refinement that ensures $y^+ \leq 1$ at the Reynolds number used for the computation $Re = 2.16 \times 10^6\ \mathrm{m}^{-1}$.

Solutions computed on the grid setups obtained with a mask-based grid assembly (see Fig. 6) and the patch assembly algorithm (see Fig. 7) are compared with results obtained on a coincident mesh (see Fig. 8). See Fig. 9 for the flow computed on the coincident mesh. On Figs. 10 and 11, a comparison of the results on each grid configuration

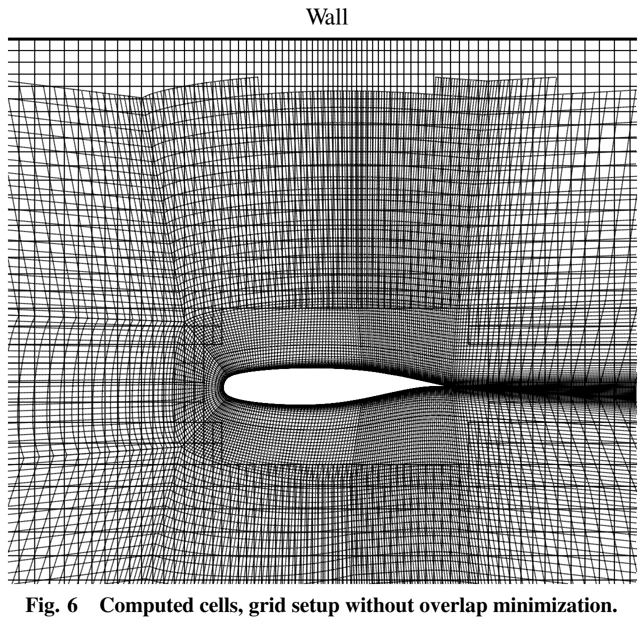


Fig. 6 Computed cells, grid setup without overlap minimization.

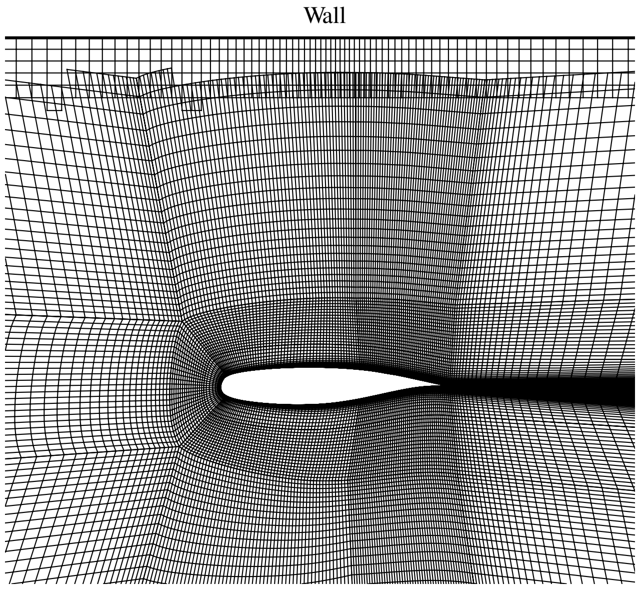


Fig. 7 Computed cells, patch assembly grid setup.

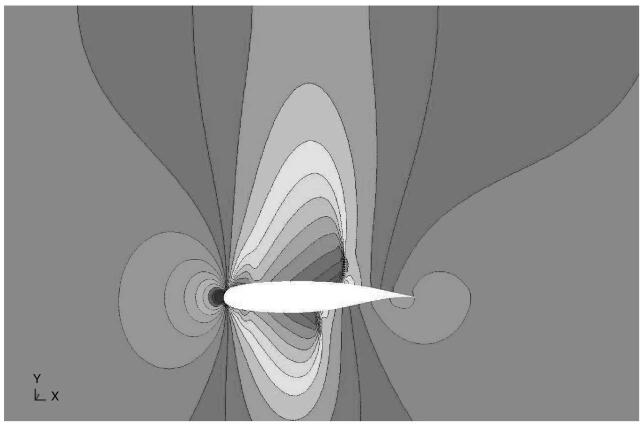


Fig. 9 Contours of pressure coefficient obtained on the coincident mesh for the steady computation.

is presented. The two conclusions of these computations are as follows:

1) The grid setup produced by the patch assembly algorithm is valid. Flow computation gives the same results as on other grid setups.

2) The type of grid and the chimera grid setup has no influence on the pressure computed on the airfoil skin (see Fig. 11). In particular, the same results are obtained on the overset mesh and on the coincident mesh.

3) The chimera grid setup does not seem to affect the convergence of the computation (see Fig. 10): residuals curves on both chimera grid setups are similar.

2. *Unsteady Flow Computations*

Unsteady flow computations were done on the NLR7301 case. For these computations, the airfoil was excited by a forced sinusoidal pitching motion around an axis located at 25% of the airfoil chord. The reduced frequency of this motion was $k = 0.21$ and its amplitude was 2 deg. Unsteady Reynolds-averaged Navier–Stokes equations were solved on each grid setup using a dual-time stepping scheme [41]. The number of physical time steps computed per period was set to 35.

As one of the motivations of these computations was to investigate the influence of the grid setup on the convergence speed, a number of subiterations were tested ranging from 20 to 180 to investigate if fewer subiterations could be used with the overlap between the chimera grids reduced.

The pitching motion of the airfoil is obtained by deforming the mesh. On the chimera meshes, only the airfoil mesh is deformed. The harmonics of the generalized aerodynamic force acting on the airfoil

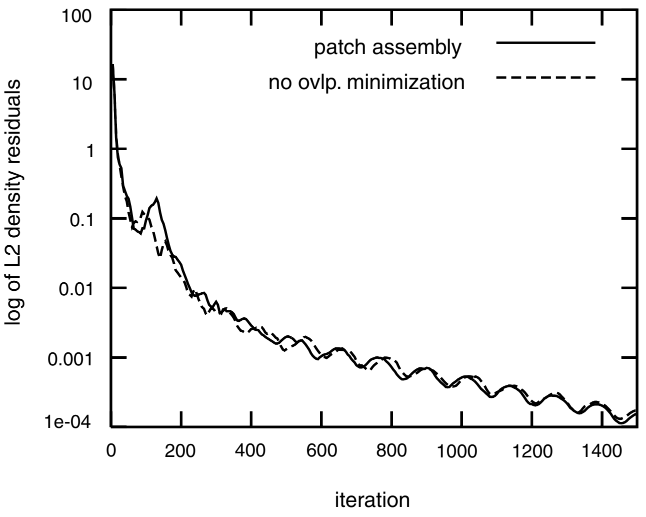


Fig. 10 Comparison of density residuals for the computations on the two chimera grid setups.

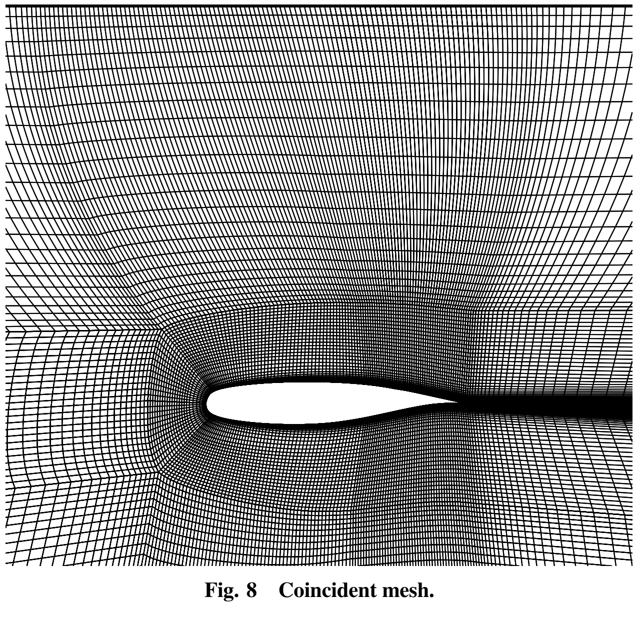


Fig. 8 Coincident mesh.

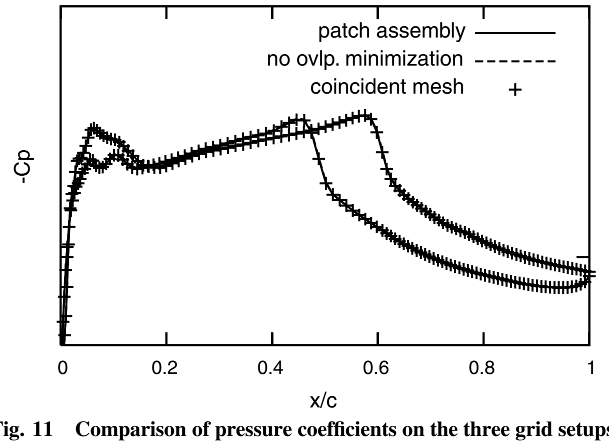


Fig. 11 Comparison of pressure coefficients on the three grid setups.

were computed (generalized force relative to the pitching motion). Its mean value, its complex module, and argument are presented, respectively, in Figs. 12–14 for each number of subiterations tested. The following points were observed:

1) The grid setup has an influence on results: a difference between the force computed on the grid setup without overlap minimization and on the patch assembly grid setup can be observed.

2) The results obtained on the patch assembly setup are closer to the results on the coincident mesh than those obtained on the grid setup without overlap minimization.

3) The grid setup does not seem to have any influence on the number of subiterations required to obtain converged values of forces acting on the airfoil. In particular, using grid setups with minimum overlap does not reduce the computational cost of an unsteady computation by reducing the numbers of subiterations performed.

4) The difference of results on the grid setups only appears for the unsteady computations (see Figs. 12–14).

The second point shows a favorable influence of the grid setup produced with the patch assembly algorithm on the computation result.

The differences between the unsteady computations on the two overlapping grid setups can be explained by the difference of refinement between the airfoil mesh and the wall mesh. The flow structures, like the shock wave over the airfoil, move differently in the two meshes while the airfoil is pitching. In the region of overlap, where these structures are computed in both meshes, these structures can have slightly different positions. This can lead to bad results on the grid setup without overlap minimization which has a large overlap between the elementary grids.

The location of the interpolated points in the wall mesh also certainly has an influence on the difference of results. In the grid setup without overlap minimization, the interpolation in the wall mesh is done in the region where the shock wave over the airfoil is located, whereas in the patch assembly grid setup, the interpolation is done in a region of weakest gradients. This can explain why the results on the patch assembly grid setup are closer to those of the coincident mesh.

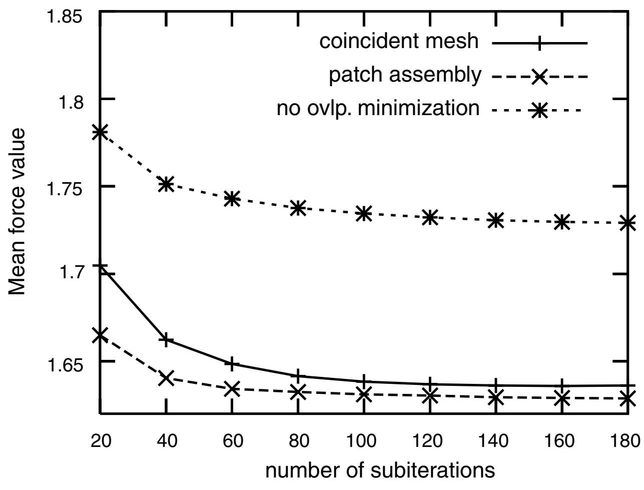


Fig. 12 Mean value of the force on the airfoil.

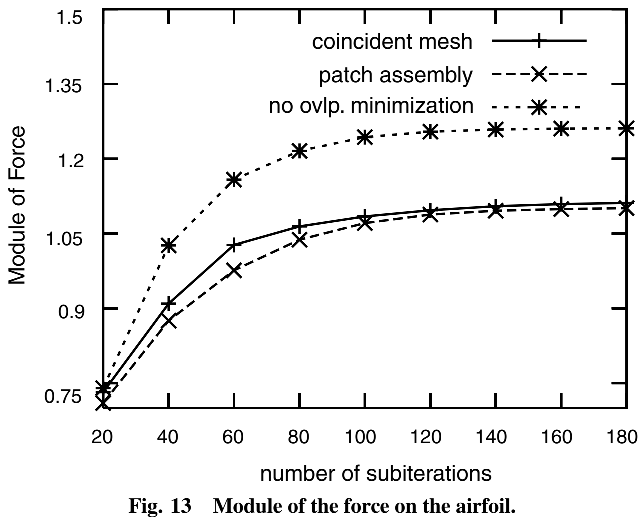


Fig. 13 Module of the force on the airfoil.

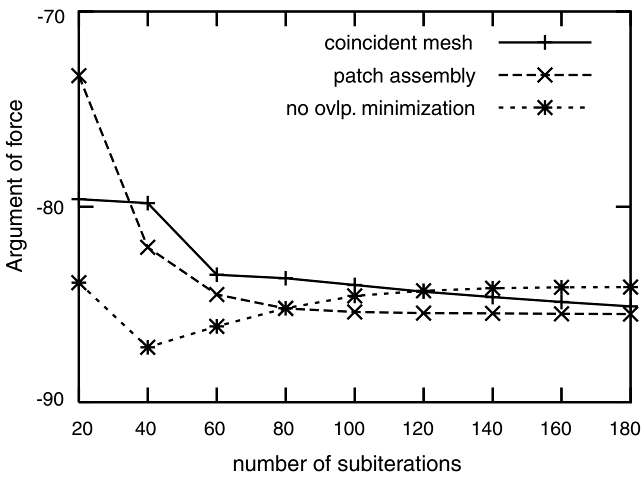


Fig. 14 Argument of the force on the airfoil.

B. Aileron on a Wing–Body Configuration

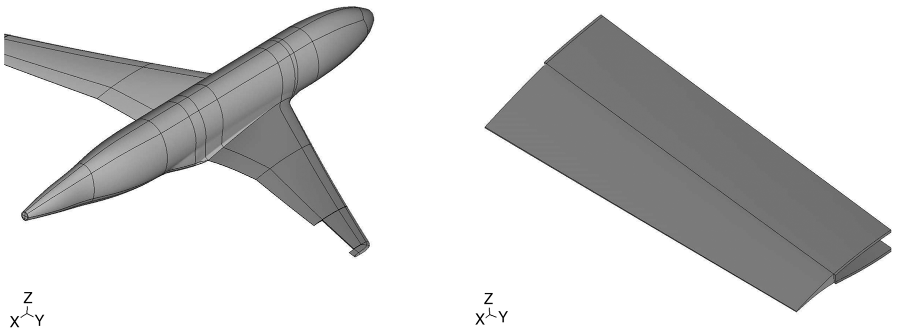
The patch assembly algorithm was applied to a case where the chimera technique is used to add an aileron to a wing–body configuration.

The background grid contains the wing–body (Fig. 15a) mesh and the patch grid contains the aileron (Fig. 15b) mesh. On this configuration, the grid assembly generated by the patch assembly algorithm was compared to a grid setup without overlap minimization, generated using a mask defined using the aileron skin. The grid setups are presented Figs. 16a and 16b for the patch assembly and Figs. 17a and 17b for the grid setup without overlap minimization. In these figures, the blanked cells have been hidden. It can be clearly seen that the overlap achieved with the patch assembly algorithm is smaller than with the mask-based approach. Moreover, setting geometrical masks on this configuration is difficult because of the small gaps between the wing and the aileron.

The results of the flow computation are presented in Figs. 18a and 18b. The chimera preprocessing and the flow computation were done in parallel on 16 processors. The same distribution of mesh blocks was used for both chimera preprocessing and flow computation. The distribution chosen favors the load balancing of the flow computation rather than that of the chimera preprocessing. The mesh has approximately 15×10^6 cells. With the patch assembly algorithm, almost 9×10^5 cells were blanked in the background grid. This shows the capability of the patch assembly algorithm to deal with big industrial meshes on large, distributed memory parallel computers.

The chimera preprocessing took almost 20 min with the patch assembly algorithm and 5 min with the mask-based blanking. This is not surprising because the patch assembly algorithm performs many more operations than the basic mask-based grid assembly process. However, the good performance of the mask-based blanking has to be balanced by the time spent by the user to iteratively set masks manually. With the patch assembly algorithm, most of the time was consumed by the search of interpolatable cells in the background grid.

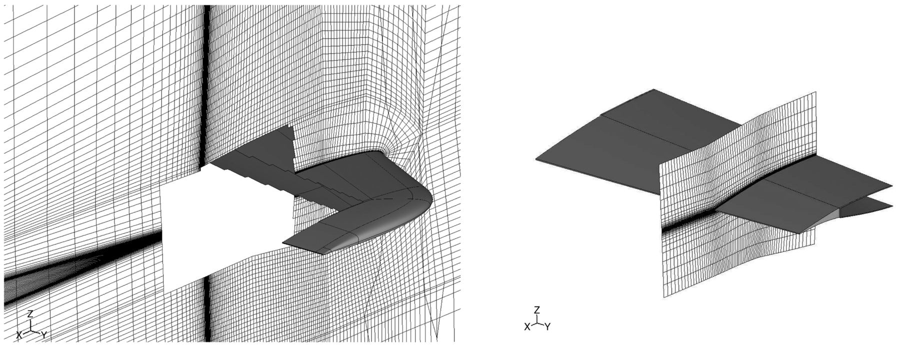
For the flow computation, the same physical model and the same numerical techniques as for the NLR7301 case were used. The Mach number is equal to 0.84 and the Reynolds number to 2.83×10^6 m^{-1}. The flow computation took around 10 h. The same computation time



a) Background: a wing-body configuration

b) Patch: isolated aileron

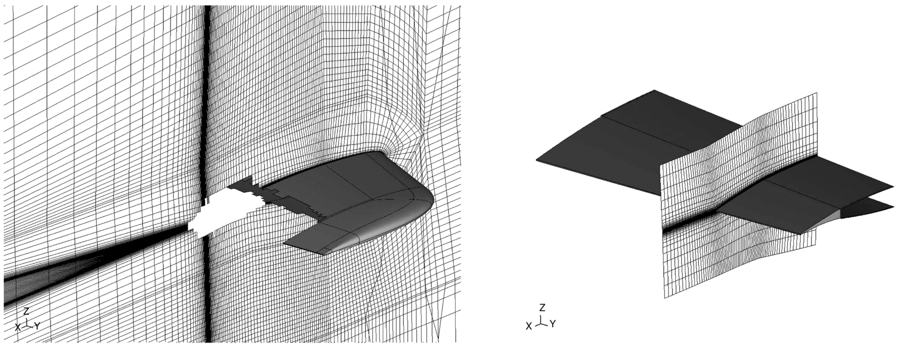
Fig. 15 Background and patch parts.



a) Background mesh

b) Patch mesh

Fig. 16 Grid setup achieved with the patch assembly algorithm.



a) Background mesh

b) Patch mesh

Fig. 17 Grid setup without overlap minimization achieved with the mask defined on the aileron skin.

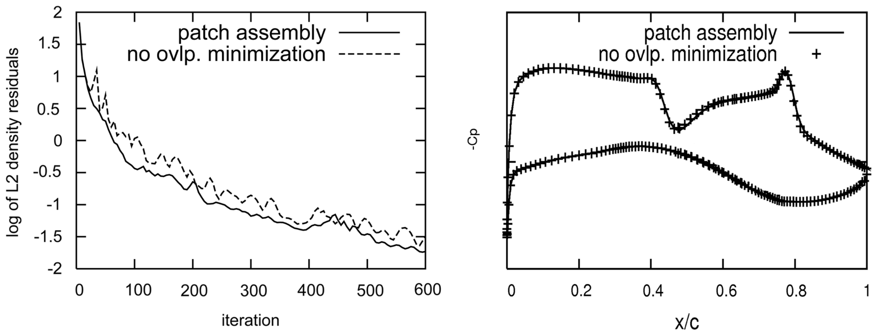


Fig. 18 Comparison of results on grid setups generated with patch assembly and with masks without overlap minimization.

is obtained on both grid setups because the number of interpolated cells is almost identical.

For the whole calculation (chimera preprocessing and flow computation), the use of the patch assembly algorithm increased the computation time by less than 5%. Figure 18b shows that the results obtained on the two grid setups are very close. These curves of pressure coefficients are typical of a transonic wing flow. A difference can be noted on the convergence curves obtained in Fig. 18a: the convergence obtained on the grid with a minimized overlap is slightly better; the residual curve is smoother and less oscillatory. The use of the patch assembly algorithm has a good influence on the computation convergence.

IV. Conclusions

A new chimera grid assembly algorithm has been proposed and validated on two different configurations. The main advantage of this algorithm is that it is fully automated and efficient because it ensures a minimum overlap between overlapping domains. Good results have been obtained with the grid setups generated by the patch assembly algorithm when compared to grid setups without overlap minimization generated with a more conventional approach based on masks. This comparison shows the significance of minimizing the overlap region after the blanking of interior cells to obtain good grid setups.

Further work could be, for example, to make the patch assembly algorithm able to deal with grid setups with several levels of patch grids to enlarge the range of use of the method or to study the influence of the chimera grid setup on a wider range of applications.

Acknowledgments

This work is part of a thesis research funded by Airbus and done in close collaboration with European Center for Research and Advanced Training in Scientific Computation (CERFACS). The author wishes to thank all the people of the Airbus aerodynamic methods and tools team in Toulouse. The elsA developers from ONERA and CERFACS, and especially Bjoern Landmann, are gratefully acknowledged for the fruitful discussions about the chimera technique.

References

[1] Steger, J. L., Dougherty, F. C., and Benek, J. A., "A Chimera Grid Scheme," *Adavances in Grid Generation*, Vol. 5, edited by K. N. Ghia, American Society of Mechanical Engineers, New York, 1983, pp. 59–69.

[2] Schwarz, T., "The Overlapping Grid Technique for the Time Accurate Simulation of Rotorcraft Flows," *31st European Rotorcraft Forum*, Council of European Aerospace Societies Paper, Sept. 2005.

[3] Benoit, C., Jeanfaivre, G., and Canonne, E., "Synthesis of ONERA Chimera Method Developed in the Frame of CHANCE Program," *31st European Rotorcraft Forum*, Council of European Aerospace Societies Paper, Sept. 2005.

[4] Lijewski, L. E., and Suhs, N., "Time-Accurate Computational Fluid Dynamics Approach to Transonic Store Separation Trajectory Prediction," *Journal of Aircraft*, Vol. 31, No. 4, 1994, pp. 886–891. doi:10.2514/3.46575

[5] Bonet, J., and Peraire, J., "An Alternating Digital Tree (ADT) Algorithm for 3D Geometric Searching and Intersection Problems," *International Journal for Numerical Methods in Engineering*, Vol. 31, No. 1, 1991, pp. 1–17. doi:10.1002/nme.1620310102

[6] Noack, R., and Kadanthot, T., "An Octree Based Overset Grid Hole Cutting Method," *Proceedings of the 8th International Conference on Numerical Grid Generation in Computational Field Simulations*, Sandia National Labs., South Lake Tahoe, CA, 2009, pp. 783–792.

[7] Dougherty, F. C., "Development of a Chimera Grid Scheme with Applications to Unsteady Problems," Ph.D. Thesis, Stanford Univ., Stanford, CA, 1985.

[8] Wang, Z. J., and Parthasarathy, V., "A Fully Automated Chimera Methodology for Multiple Moving Body Problems," *International Journal for Numerical Methods in Fluids*, Vol. 33, No. 7, 2000, pp. 919–938. doi:10.1002/1097-0363(20000815)33:7<919::AID-FLD944>3.0.CO;2-G

[9] Meakin, R., "Object X-Rays for Cutting Holes in Composite Overset Structured Girds," *15th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2001-2537, June 2001.

[10] Noack, R. W., "A Direct Cut Approach for Overset Hole Cutting," *18th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2007-3835, June 2007.

[11] Meakin, "A New Method for Establishing Intergrid Communication Among Systems of Overset Grids," *Proceeding of the 10th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 91-1586, June 1991.

[12] Chiu, I. T., and Meakin, R., "On Automating Domain Connectivity for Overset Grids," *33rd Aerospace Sciences Meeting and Exhibit*, AIAA Paper 1995-854, Jan. 1995.

[13] Shih, T., "Overset Grids: Fundamentals and Practical Issues," *20th AIAA Applied Aerodynamics Conference*, AIAA Paper 2002-3259, June 2002.

[14] Cho, K., Kwon, J., and Lee, S., "Development of a Fully Systematized Chimera Methodology for Steady/Unsteady Problems," *Journal of Aircraft*, Vol. 36, No. 6, 1999, pp. 973–980. doi:10.2514/2.2538

[15] Yan, C., Li, T., and Xie, L., "Enhancements of the Cut Paste Algorithm in Overlapping Grid Generation," *Journal of Aircraft*, Vol. 77, No. 2, 2007, pp. 672–674.

[16] Wey, T. C., "Development of a Mesh Interface Generator for Overlapped Structured Grids," *12th AIAA Applied Aerodynamics Conference*, AIAA Paper 91-1586, 1994.

[17] Lee, Y., and Baeder, J., "Implicit Hole Cutting: A New Approach to Overset Grid Connectivity," *16th AIAA Computational Fluid Dynamics Conference*, AIAA Paper 2003-4128, June 2003.

[18] Liao, W., Cai, J., and Tsai, H. M., "A Parallel, Multigrid Overset Grid Flow Solver Using Implicit Hole Cutting," *22nd Applied Aerodynamics Conference and Exhibit*, AIAA Paper 2004-5072Aug. 2004.

[19] Sitaraman, J., Floros, M., Wissink, A. M., and Potsdam, M., "Parallel Unsteady Overset Mesh Methodology for a Multi-Solver Paradigm with Adaptive Cartesian Girds," *26th AIAA Applied Aerodynamics Conference*, AIAA Paper 2008-7177, Aug. 2008.

[20] Cai, J., Tsai, H., and Liu, F., "An Overset Grid Solver for Viscous Computations with Multigrid and Parallel Computing," *16th AIAA computational fluid dynamics conference*, AIAA Paper 2003-4232, June 2003.

[21] Cai, J., Tsai, H., and Liu, F., "A Parallel Viscous Flow Solver on Multi-Block Overset Grids," *Computers and Fluids*, Vol. 35, No. 10, 2006, pp. 1291–1301.

[22] Suhs, N., Dietz, W. E., Rogers, W. S., Nash, S., and Onufer, J., "Pegasus User Guide," Ames Research Center Technical Rept., Oct. 2001.

[23] Suhs, N. E., Rogers, S. E., and Dietz, W. E., "PEGASUS 5: An Automated Pre-Processor for Overset-Grid CFD," *AIAA Journal*, Vol. 41, No. 6, 2003, pp. 1037–1045. doi:10.2514/2.2070

[24] Chesshire, G., and Henshaw, W. D., "Composite Overlapping Meshes for the Solution of Partial Differential Equations," *Journal of Computational Physics*, Vol. 90, No. 1, 1990, pp. 1–64. doi:10.1016/0021-9991(90)90196-8

[25] Petersson, N. A., "An Algorithm for Assembling Overlapping Grid Systems," *Journal of Scientific Computing*, Vol. 20, No. 6, 1999, pp. 1995–2022. doi:10.1137/S1064827597292917

[26] Noack, R. W., and Boger, B. A., "Improvements to SUGGAR and DiRTlib for Overset Store Separation Simulations," *47th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, AIAA Paper 2009-340, Jan. 2009.

[27] Togashi, F., Ito, Y., Nakahashi, K., and Obayashi, S., "Extensions of Overset Unstructured Grids to Multiple Bodies in Contact," *Journal of Aircraft*, Vol. 43, No. 1, 2006, pp. 52–57. doi:10.2514/1.540

[28] Slotnick, J.-P., Kandula, M., and Buning, P. G., "Navier-Stokes Simulation of the Space Shuttle Launch Vehicle Flight Transonic Flowfield Using a Large Scale Chimera Grid System," *12th AIAA Applied Aerodynamics Conference*, AIAA Paper 1994-1860, June 1994.

[29] Chan, W. M., "Overset Grid Technology Development at NASA Ames Research Center," *Computers and Fluids*, Vol. 38, No. 3, 2009, pp. 496–503. doi:10.1016/j.compfluid.2008.06.009

[30] Fillola, G., Carrier, G., Rodde, A.-M., and Dor, J.-B., "Experimental Study and Numerical Validation of Flows Around Wing Control Surfaces," *International Council of the Aeronautical Sciences Conference 2006*, German Society for Aeronautics and Astronautics, Bonn, Germany, Sept. 2006.

[31] Schwarz, T., Ein Blockstrukturiertes Verfahren zur Simulation der Umströmung Komplexer Konfigurationen, Ph.D. Thesis, Inst. für Aerodynamik und Strömungstechnik, Brunswick, Germany, 2005.

[32] Nakahashi, K., Togashi, F., and Sharov, D., "Intergrid-Boundary Definition Method for Overset Unstructured Grid Approach," *AIAA Journal*, Vol. 38, No. 11, 2000, pp. 2077–2084. doi:10.2514/2.869

[33] Chan, W., Gomez, R., Rogers, S., and Buning, P., "Best Practices in Overset Grid Generation," *32nd AIAA Fluid Dynamics Conference*, AIAA Paper 2002-3191, June 2002.

[34] Cambier, L., and Gazaix, M., "elsA: An Efficient Object-Oriented Solution to CFD Complexity," *40th AIAA Aerospace Science Meeting and Exhibit*, AIAA Paper 2002-0108, Jan. 2002.

[35] Delbove, J., "Unsteady Simulations for Flutter Prediction," *Computational Fluid Dynamics 2004*, Springer, Berlin, 2006, pp. 205–210.

[36] Spalart, P. R., and Allmaras, S. R., "A One-Equation Turbulence Transport Model for Aerodynamic Flows," *30th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA Paper 92-0439, Jan. 1992.

[37] Jameson, A., Schmidt, W., and Turkel, E., "Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes," *AIAA 14th Fluid and Plasma Dynamic Conference*, AIAA Paper 81-1259, June 1981.

[38] Yoon, S., and Jameson, A., "An LU-SSOR Scheme for the Euler and Navier-Stokes Equations," *AIAA 25th Aerospace Sciences Meeting*, AIAA Paper 87-0600, Jan. 1987.

[39] Juvigny, X., Canonne, E., and Benoit, C., "Multigrid Algorithms for the Chimera Method," *42nd Aerospace Sciences Meeting & Exhibit*, AIAA Paper 2004-758Jan. 2004.

[40] Henshaw, W., "On Multigrid for Overlapping Grids," Lawrence Livermore National Lab. Technical Rept., May 2004.

[41] Jameson, A., "Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings," AIAA Paper 91-1596, 1991.