ERF2013-075

COMPARISON BETWEEN SLIDING AND CHIMERA GRIDS

Mikolaj Jarkowski, Mark Woodgate and George N. Barakos

CFD Laboratory, Department of Engineering
University of Liverpool, L69 3GH, U.K.
http://www.liv.ac.uk/flightscience/PROJECTS/CFD/ROTORCRAFT/index.htm

Abstract

The Overset Mesh Method (chimera) is popular in rotorcraft research, since the use of multiple, non-matching grids makes the Computational Fluid Dynamics simulations of bodies in relative motion much simpler. Consequently, the relative motion between the helicopter blades and fuselage can be accurately accounted for. In this paper, the method for treating overset grids within CFD codes is presented and compared against other methods like sliding planes and fully matched grids.

The proposed method is based on hierarchy of overset, non-matching grids, whose cells are automatically identified as computational or non-computational and localised with respect to all grids they overlap with. The efficiency of the method relies on the fast, *cell-in-solid* detection, based on combined *Bounding Box* - *Ray Tracing* method, the hierarchical, multi-step approach, for the overset mesh localisation and the use of a tree search. Due to the high efficiency of the algorithm the search for overlapping cells can be carried out on-the-fly, during time-marching of the unsteady, implicit CFD solver.

The chimera method algorithm will be described in details, together with its extensions, that improve the localisation of the solution as well as allow for the computation of the cases with intersecting solid boundaries.

Nomenclature

LP	Localisation Pre-processing
OMS	Overset Mesh Search
EAL	Exact Arithmetics Library
GCS	Global coordinate system
MVBB	Minimum Volume Bounding Box
AABB	Axis Aligned Bounding Box
RT	Ray Tracing
TCS	Transformed coordinate system
x_i	Coordinates in the i^{th}
ξ_k	Coordinates in the k^{th}
I	Second moment of area matrix
m_i	Mass of the i^{th} grid cell, assuming the density of $1\frac{kg}{m^3}$
$\vec{r_i}$	Vector from the origin of the GCS to the centroid of the i^{th} grid cell
$\vec{F_i}$, $\vec{F_v}$	Inviscid and viscous fluxes
$\mathbf{R}_{i,j,k}$	Flux residuals at cell (i,j,k)
$\vec{S}$	Source term compensating for inertial effects
$V(t)$	Time dependent control volume
$\vec{u}_h$	Local velocity field in the rotor-fixed frame of reference
$\mathbf{w}_{i,j,k}$	Discretised conserved variables vector
$\vec{\mathbf{w}}$	Conserved variables vector
ρ	Air density
$\vec{\omega}$	Rotor rotational speed

1 Introduction

For the purposes of this work, the Helicopter Multi-Block CFD solver (HMB2) [1] is used. The chimera method of the HMB2 is based on hierarchical hole cutting. Further extensions to the chimera method are presented, which expand the flexibility of the solver. The main modifications of chimera method consist in the utilisation of semi-infinite ray tracing method for a *point-in-solid* detection and wall-distance based boundary layer correction, that allows for the use of embedding solids - intersecting solid surfaces belonging to different grid levels. The modifications were possible due to the data-structure of HMB2 that is flexible enough to accommodate extensions to the solver without major re-writing of the core functions [2].

1.1 Introduction to Localisation Methods

The grids used in HMB2 are multi-block, structured and body-fitted with hexahedral elements, numbered as presented in Fig. 2a. It is assumed that each cell face consists of four triangles with a common apex located on the face centroid (Fig. 2b). Each of the 24 triangles (4 on each cell face) forms a base of a tetrahedron. A hexahedral cell is replaced by a collection of 24 tetrahedra with a common apex at the cell centroid. This operation is consistent in terms of mesh volume as the position of each wall centroid is independent of the side it is calculated from (inside or outside). This cell-split operation is commonly used within the current search algorithm, as any *point-in-cell* check is based on tetrahedra. Moreover, unlike a hexahedron, a tetrahedron has a simple linear transformation into a normalised shape (Eq. 1):

$$x_i = {x_i}^0 + \sum_{k=1}^{3} \left({x_i}^k - {x_i}^0\right) \cdot \xi_k \tag{1}$$

dimension of the GCS dimension of the TCS

where: subscripted indices (1, 2 and 3) represent dimensions in orthogonal coordinate system and superscripted indices (0, 1, 2, 3) represent one of four vertices of a tetrahedron. The transformation into a normalised shape is used when determining the interpolation weights for inter-grid communication. This is explained in the *Overset Mesh Search* section.

The chimera localisation methods identify the cells that require interpolated flow information from the grid they overlap with, and provide interpolation weights. Figure 1a, shows the chimera case with two grids, background grid (black) and the inner grid (foreground grid, around the aerofoil). The outer boundary of the inner grid will herein after be referred to as chimera boundary, and its halo mesh (not shown in the Figure) will require interpolated flow data (from the background grid). Figure 1b, shows the flags on the background grid and the cells shown in light grey (called interpolation cells) also require interpolated flow data (from the foreground grid - to provide data exchange in both directions).

The chimera functions in HMB2 are divided into two major groups. Localisation Pre-processing (LP) is the first of them and consists of a simple set of geometric operations that are always conducted before the second step, the Overset Mesh Search (OMS).The main idea used within LP is the construction of approximate Minimum Volume Bounding Boxes (MVBBs). MVBBs are cuboids that completely bound a given shape and occupy the minimum possible volume. In HMB2, MVBBs are computed based on the second moment of area matrix (Eq. 2), whose eigenvectors provide direction vectors of the principal axes of inertia. The ranges of the MVBBs are determined by rotating the geometric entity (*e.g.* mesh block) so that its principal axes of inertia match the Global Coordinate System (GCS) and by finding minimum and maximum values of X, Y and Z variables. The approximate MVBBs in the HMB2 do not always occupy the minimum possible volume, but the approximation is good enough for LP.

The second moment of area matrix is defined as:

$$I = \sum_i m_i \cdot \vec{r_i} \otimes \vec{r_i} \tag{2}$$

where $\vec{r_i}$ is the vector representing the coordinates of the i^{th} cell centroid, with respect to the block's centre of gravity, and m_i represents its volume.

Another type of a bounding box (used within OMS) is the Axis Aligned Bounding Box (AABB). It poses a cuboid that bounds completely the geometric entity it is associated with, and is aligned with the GCS axes. Effectively, an AABB can be defined by two points in space: $P_{\min} = (x_{\min},\, y_{\min},\, z_{\min})$ and $P_{\max} = (x_{\max},\, y_{\max},\, z_{\max})$.

Additionally, two main computational geometry tools are used during the OMS: a range-tree and an Exact Arithmetics Library (EAL). A range-tree is a structure built of a 3D set of points, and efficiently determines, which of the points are located in a given bounding box. Within the scope of this work, the AABB is used for querying the tree. In Fig. 4 a 2D example of a range tree is shown. The tree is constructed of black dots, and we want to determine, which of these are potentially located inside the shaded area. The tree becomes the shaded area's axis-aligned bounding box as an input (rectangle) and returns all the points inside the rectangle. This search is very efficient, following $k\cdot$ log(n) computational cost (where k is a number of points in the tree localised against n volumes/areas).

The second tool (EAL) checks point positions with respect to a plane (3D) or a line (2D). In general, the position of a point P x_P, y_P, z_P with respect to a plane defined by three points A, B, C can be verified by checking the sign of (Eq. 3):

$$\det \begin{vmatrix} x_A - x_P & x_B - x_P & x_C - x_P \\ y_A - y_P & y_B - y_P & y_C - y_P \\ z_A - z_P & z_B - z_P & z_C - z_P \end{vmatrix} \tag{3}$$

The EAL computes equation 3 with the maximum possible machine precision. The details on this method and assurance for the correct sign of the determinant can be found in [3]. The use of EAL is important, because it guarantees that point localisation with respect to cells is unambiguous *i.e.* any point can only be located in one cell of a level, it is localised against. What is more, a volume of a cell, in extreme cases, can be of the order of $1.0 \cdot 10^{-18}$, so high precision computational geometry tools are required.

1.2 Localisation Pre-Processing (LP)

The load balancing, necessary to obtain high parallel efficiency is based on the CFD computational cost (the chimera localisation cost is not taken into account) and the composite grid is treated as if it was a single-level, matched grid. As a result, in case of parallel execution, each processor stores blocks from different sub-grids (herein after referred to as levels - due to the hierarchical approach). Consequently, considerable data exchange is required for the localisation of cells of different levels. In order that the number of searches is limited to minimum, a Localisation Pre-processing (Fig. 5, left top, bounded with a dashed line) is used. After this operation, each block is given a list of blocks of a different level, it has to be localised against. The localisation pre-processing consists of three steps: i) generation of MVBBs, ii) a block-to-block search operation, iii) node-in-solid MVBB search operation. The first step generates MVBBs for all the blocks (block MVBBs) in a composite grid plus a MVBB for each boundary, tagged as solid wall (solid MVBBs). The second operation checks which blocks potentially overlap, based on the MVBB information. The third step utilises solid MVBBs and flags all the nodes, which have been localised inside solid MVBBs of any other level as *node-in-solid-mvbb*.

In Fig. 5, (inside the dashed box) the flowchart of the localisation pre-processing is shown. Minimum Volume Bounding Boxes (MVBBs) are constructed for each block. The halo mesh of the chimera boundaries (see Fig. 6, both first and second layer) needs to be included in the MVBB ranges. Halo cells in the *Chimera Interface* includes the cells that require interpolated information, so it has to be determined which blocks of other levels they overlap with. Halo cells of any other block face (not associated with chimera boundary) are excluded. Experience with HMB2 proved, that the best approximation of the MVBB is obtained when only the first layer of chimera boundary halo cells is used in determining the principal axes of inertia. The extrapolation of the second layer of halo mesh could generate cells of negative volume, especially on grids with high cell aspect ratios. Neglecting the second layer of halo mesh at this stage will not lead to exclusion of any cells from the bounding box, as the second row of halo mesh for chimera boundaries is taken into account at the stage of range definition. MVBB generation is a sequential operation (each processor generates MVBB for the blocks it stores) and is followed by an MPI operation, distributing all MVBBs to all processors. Figure 7 shows an example of a multi-block topology and the associated MVBB. The halo mesh of the chimera boundaries (both first and second layer) is included in the MVBB ranges.

The operations described above refer to the geometrical features of the blocks that are treated like three-dimensional volumes with a uniform density distribution.

Additional bounding boxes are also created for all solid bodies in the grid. Unlike blocks, solid surfaces have no volume, so the principal axes of inertia have to be determined based on the surface information only. As a result, solid bounding boxes do not strictly take the minimum possible volume, and the MVBB approximation is worse than for blocks, but sufficient for solid hole cutting. The generation of solid MVBBs, however, is not sequential, as parts of one solid surface may be distributed between different processors. MVBBs of solid surfaces are used for flagging *nodes-in-solid*. An example is shown in Fig. 8, where all the points of the background grid inside the aerofoil shape are marked as *nodes-in-solid-MVBB*.

In the next step, operations are performed that identify all block-to-block overlaps between blocks of different levels. This is done by checking if the nodes of each block of one level are located inside the MVBB of any blocks belonging to another level (the concept of mesh *level* is presented in Fig. 7, where two multi-block, overlapping topologies are hierarchically ordered). For efficiency, only the external block nodes are taken into account. This algorithm is very efficient and data parallel, however, it is optimistic in the generated block-to-block overlap information. Consequently, some of the block overlaps that are to be tested for cell overlap will produce no results.

The overlap information needs to be symmetric, *i.e.* if block A overlaps with block B, block B should also overlap with block A. However, if a block is completely embedded inside another block, the output of the procedure described above will be non-symmetric (a bounding box of the embedded block does not contain any external points of the outer block). Therefore, a cleanup operation is required to obtain a fully symmetric overlap information.

$$\widehat{O} = O \vee O^T$$

where O is a bitwise matrix containing the overlap information (0 if non-overlapping, one if overlapping) for all the blocks in the grid. The size of the matrix is therefore equal to total number of blocks in the grid (width and length) and $\widehat{O}_{nblt\,x\,nblt}$ is the full symmetric bitwise overlap information, and the size subscript $_{nblt}$ refers to the total number of blocks in the overset grid. In case of a parallel execution, this operation is done on a root processor.

Since memory in HMB2 is allocated per block, finding all the blocks with potential overlap provides a mechanism for allocating the minimum required memory, as well as restricting the necessary searches for the overset mesh functionality at cell-level.

At the end of the LP, all mesh nodes are localised with respect to all solid MVBBs of other levels. Solid MVBBs store the information relating to the grid level they belong to and the solid shape they are associated with. Each solid MVBB is a rough approximation of the solid shape, so some nodes, which are outside of the solid shape, are flagged as *in-solid* within this search (see Fig. 8). For those nodes, the flag will change at the later stage. Any node which has an *in-solid* flag, and at the same time, has been located inside any cell of the mesh, the solid MVBB was associated with, will lose the *in-solid* flag (Fig. 9).

1.3 Overset Mesh Search (OMS)

1.3.1 Methodology.

The cell localisation process has two main objectives: (i) to identify the type of each cell (Fig. 1), and (ii) to determine the interpolation stencils for the cells that require interpolated information. Cell flags are assigned based on node position with respect to the other levels of the composite grid (Fig. 12, this will be explained at the end of this section).

At the end of this step, each cell is marked as a: *"cell-in-solid"*, *"hole-cell-in-fluid"*, *"hole-cell-with-interpolated solution"*, *"fringe"* or *"normal-cell"*. This extra information can be stored in a file for the HMB2 solver to read, or can be calculated on-the-fly as the HMB2 solver is executed. It should be noted that HMB2 merges cell-flag information into three classes: normal cells, (*"fringe"* and *"normal-cell"*), holes, (*"cell-in-solid"*, *"hole-cell-in-fluid"* and *"hole-cell-with-interpolated solution"*) and finally interpolation (*"hole-cell-with-interpolated solution"*) which are the only cells that require extra information associated with them.

Throughout the localisation process, the geometric entities (nodes, cell centroids) are localised within cells belonging to the grids of different levels. Consequently, only the information specifying by which cell the currently localised geometric entity is bounded is significant, rather than the nearest cell information. It has to be noted that finding the nearest cell centroid does not guarantee that the cell bounds currently localised point. In the case of a grid with high cell aspect ratios, the centroid of the cell that bounds a point may not be the nearest one. Figure 10 presents such a case.

The OMS consists of three major steps:

1. Localisation of nodes and centroids with respect to all potentially overlapping blocks.
2. Cell flagging, based on the node flags from the previous step.
3. Computation of the interpolation weights for all cells requiring interpolated data.

During OMS, both node flags, and cell flags are used. At the end of the search, however, only the cell flags are required. Node flags are only required as auxiliary attributes, for the correct determination of *fringe* cells. Node flags are not required for the CFD part, and after the cell flags are determined, they can be freed from memory.

The aim of the OMS is to establish the necessary inter-grid communications. For the HMB2 solver, the inter-grid communication information consists of the list of deactivated cells, the list of cells requiring interpolated data and the associated interpolation weights. The chimera search only includes computational geometry operations, based on the grid points, topology data, information provided in the input file and the Localisation Pre-processing output.

In the first step, the OMS algorithm performs hole cutting, *i.e.* it detects the cells of the mesh where flow solution will not be computed. Hole cutting is based on mesh hierarchy. The hierarchy of grids in HMB2 is defined by the user, via an input file. The hole cutting is done in two parts: the nodes are localised with respect to other level grids and then the cells are flagged based on the flags of their vertices. Node flags applied within this step indicate that a node has been localised in any cell of a lower level, any cell of a higher level, or it is a normal node. Flags may be appended, so any node can be localised in a lower level cell and in a higher level cell. Additionally, some of the nodes in the mesh may already be flagged as *node-in-solid*. All the information that needs to be associated with the nodes is limited to one of the four flags mentioned above. Nodes do not need to store any information about the index of the cell, they have been found inside.

The hole cutting procedure uses a range tree. The tree is created for all blocks, that potentially overlap with any other blocks on other levels and consists of the coordinates of block nodes and the coordinates of cell centroids. The tree also includes the first layer of halo mesh for nodal coordinates and two rows of halo mesh for cell centroid coordinates.

The tree search is only conducted for pairs of blocks identified as potentially overlapping in a block-to-block search. As the information about block intersections does not specify their extent, it is required that each tree is checked against all cells of all blocks, the currently localised block potentially overlaps with. For clarity, two terms will be used in the description of tree search: *flagged block* - the one whose nodes and cell centroids are currently localised with respect to another level, and with which the tree is associated and *flagging block* - the one whose cells are used for inquiring the tree. The *flagged block* has to be checked against the AABBs of all cells of *flagging block*. In case of a parallel execution, some overlapping blocks will be assigned to different processors. In this case, MPI data exchange is required. A tree is never sent to other processors. If required, the coordinates of the flagging block mesh are sent to the processor, which owns a flagged block and its tree.

As the tree search only localises points (nodes and centroids) inside a cell AABB, they have to be additionally, checked against the cell. For an exact and consistent *node-in-cell* check, the tetrahedra are used. For this reason, Schewchuk's *Exact Arithmetics Library* [4] is used. The library helps to determine whether a point is inside a tetrahedron, by checking the point's position with respect to the faces of the tetrahedron. If the point is located exactly on the face of a tetrahedron, it is considered to be inside of it. At the same time, it is flagged as *in-cell* and is not localised against any other cell of this level, in order that it's position is unambiguous. The library is used because it allows high precision for node (and centroid) localisation.

At the end of this operation (repeated for each block in the overlapping group), each node of the flagged level has a flag, specifying if it is a *node-in-lower-level-cell*, *node-in-higher-level-cell*, *node-in-solid-box*, or no flag (normal node). The tree is constructed both of node coordinates as well as cell centroid coordinates. Consequently, when the tree is inquired by the cell ranges of overlapping blocks of other levels, it returns the list of nodes and centroids located in the range. The centroids' coordinates are also checked with respect to cells using the EAL, in the same way as nodes. For these, however, no flags are applied at this point, but the indices of the cell, inside which they were located are stored. If the composite grid consists of more than two levels, the centroid of a chimera block can be localised inside more than one cell, since it may be located in more than one overlapping levels. In such a case, the indices of all grid levels, in which the centroid has been located, are stored. This information will only be needed for the interpolation cells, and will be ignored for normal cells and hole cells. The cell flags, however, are not known at this point of the chimera localisation, but storing the potential donor cell indices for all cells saves computational effort, as no additional tree search will be required at any later stage.

After the node localisation is completed, the cells are flagged according to the following rules (see Fig. 12):

1. Any *in-solid* node is unflagged, if it has been localised as *node-in-cell* inside of the level, the flagging solid MVBB was associated with.
2. A cell is marked as a *hole* if all 8 nodes are inside a grid of a higher level.
3. A cell is marked as a *fringe* if at least one and less than 8 nodes of a cell are inside a grid of a higher level.
4. A cell is marked as a *solid* if at least one of its nodes is flagged as *in-solid*.

At this point, the *node-in-solid* information is corrected (see point (1) above). All the *in-solid* nodes store the level number of the solid MVBB used for flagging them. An *in-solid* node looses its flag if it has been localised inside any cell of the level, the solid MVBB was associated with.

For chimera functionality additional cells have to be flagged, for which flow variables will be interpolated. As fringes are the last *layer* of computational cells, fluxes through their walls are calculated based on flow data from adjacent cells (see figure 11). This is a viscous stencil, in case of Euler model, stencil is reduced. Interpolation cells are non-active in a computational sense, but the interpolated values in their centroids are used for the calculation of the fluxes into the fringe cells. Interpolation holes are flagged recursively - each *hole* neighbouring with a fringe gets an *interpolation-hole* flag, and applies the same flag to all its neighbours. Regardless of the CFD model used (Euler or viscous), interpolation cells are always flagged based on viscous MUSCLE stencil shape. This operation finalises cell flagging.

1.4 Point-in-solid Detection

Point-in-solid detection is a difficult task, since the solid shape does not have a grid inside. The method for *point-in-solid* detection is based on the idea of Minimum Volume Bounding Box (MVBB), which is a very rough approximation of a solid shape. The idea consists in flagging mesh nodes as *in-solid-bounding-box*, that are the candidates to be flagged as *in-solid* in the later stage. Any *in-solid-bounding-box* node will not be flagged as *in-solid* only if it is localised inside any cell of the grid level, the solid MVBB is associated with. Otherwise (if it is not localised inside any cell of the grid, the solid MVBB is associated with), it is flagged as *in-solid*. The disadvantage of this method is that it imposes a restriction on an overset grid. The grid associated with a solid shape has to cover the solid MVBB of the solid shape. In other words, The MVBB of the solid shape associated with the grid cannot cut through the grid boundary. The restriction can be removed if the solid MVBB method is supplemented with additional, more precise search, *e.g.* Ray Tracing (RT).

RT combined with solid MVBB, can provide a very efficient way of *point-in-solid* detection. Therefore, the method was added to the HMB2. The method is based on semi-infinite rays, which are simple straight lines with the origin in the point, localised against a solid surface. If a ray, traced from the point of interest, does not cross the solid surface, or crosses the solid surface an even number of times, the point is not bounded by the surface If it crosses the solid surface a odd number of times, the point is inside the volume bounded by the surface (see Fig. 14).

The RT method of HMB2 utilises a range-tree [1]. Rays are not shot from the points (nodes), that have to be localised against the solid surface. The nodes on the solid surfaces are used for the generation of volume, inside which the grid nodes are localised (see Fig. 15). Boundary cells, constructed of 4 nodes on the solid surface, are then extruded along the positive Z direction. The extent of the cell extrusion can be treated as infinite, compared to the dimensions of the grid. In next step, the range-tree is used to localise all *in-solid-bounding-box* nodes of all levels not associated with the solid surface, against the semi infinite cell generated in the previous step. The points returned by the range-tree are checked against the extruded cell using the EAL [3]. If the point is localised as inside the cell, it is offset by a very small distance along the X and Y axes (less than 10 per cent of lowest cell dimension) and checked again with use of EAL. It the point is still inside the cell, the counter for number of ray intersections with a solid surface is incremented. The operation of double-check with the point offset is conducted in order that the number of ray intersections with a solid surface is unambiguous. If this operation was not conducted, a point which is located exactly between two cells could be identified as inside of both of them. Consequently, the number of ray intersections for this point with the solid surface would be invalid, and the *in-solid* search for this point would provide a false answer.

1.5 Boundary Layer Correction

In a chimera approach, the grids associated with solid elements are generated separately. The orientation of the boundary layer refinement on both grids is different, and follows the normal direction to the solid surface (see Fig. 16). The boundary layers are equally important from the point of view of the global resolution of the flow. The standard hole cutting hierarchy, therefore, has to give priority to the boundary layer correction method. As an example, the overset mesh between an aerofoil and a Gurney flap is considered.

After the grids are merged to one composite overset grid, the cells in the intersection corner, between the aerofoil and the Gurney flap do not match (see Fig.17). Additionally, The Gurney flap can be actuated in a linear way (sliding Gurney flap) and in a rotational way (swinging Gurney flap).

The challenges presented above require that the overset mesh algorithm deals with (i) *solid hole cutting in the highest level grid by the solid shape of a lower level*, (ii) *decision process, on which grid to solve in the region, where boundary layers are very close to one another* and (iii) *treatment of the cells in the boundary layer, which intersect with the solid shape of a different level*.

In the chimera method, the situation where two different solid surfaces in two different levels intersect with one another is allowed. Consequently, the cells of the highest level (Gurney flap level in this paper), which are located inside the solid surface of a lower level, have to be flagged as non-computational (see Fig. 18a). In addition, not all halo cells of the *chimera interface* boundary associated with the Gurney flap level, which are used as interpolation cells, can find a valid interpolee (they can be located inside the solid shape of a lower level). As a result, the halo cells of the *chimera interface* boundary, whose centroids are located inside of a solid shape, have to be removed from the interpolation cell list. Detection of the cells of the highest level in the solid surface of the lower level is not different from the method used in the previous version for the lower levels (based on solid MVBB). The *in-solid* flags for halo cells on *chimera boundaries* are flagged following a different algorithm. It is assumed, that halo cells on *chimera boundaries*, which are not bounded by any cell of any other level, are orphan points (interpolation points outside of the computational domain), and they are removed from the interpolation cell list. This may lead to the situation, where the halo cells on *chimera boundaries* are orphans, but the block cells directly on the chimera boundary are computational. If nothing s done at this point, there would be no rule for flux calculation through the chimera boundary. In order that each computational cell has a consistent rule for flux computation, two layers of cells right next to invalid chimera boundary is flagged as interpolation cells. There is also another type of orphan cells. They are chimera boundary cells, which are found inside other level cells, but the cells they are localised inside are flagged as non-computational. This will be discussed in the paragraphs below.

Another important issue in the boundary correction problem is to decide, which cells have to remain computational in the region, where boundary layers of two different levels may interact. The grid priority (ii) depends on the hierarchy in regions far from the boundary layer. The boundary layer thickness, for each solid shape separately, is defined by the user. This boundary layer is the region, where no hole cutting is done, unless the cell is located inside the solid shape. Typically, it is necessary to specify the boundary layer around the aerofoil/wing/rotor-blade shape only, since the gurney flap is always in the highest level. It has to be noted that in the boundary layer region, there is a multiple discretisation of the domain. It also leads to the situation where a computational cell directly neighbours with a non-computational one (computational cell of the boundary layer next to a solid-hole cell).

In order that a smooth solution can be obtained between two grids, interpolation is required. Two layers of computational cells, located directly next to the solid region are converted to interpolation cells (see Fig. 18b), in a manner that provides consistent the flux computation.

Some of the interpolation holes, however, are located in the region which is already flagged as non-computational on the background grid. Consequently, no valid value for interpolation can be found. Those interpolation cells without a valid interpolee are at this stage flagged as solid cells (see Fig. 18c). The face of a computational cell which directly neighbours with a solid cell is converted to *internal wall*. It means, that no velocity fluxes are calculated for this wall and is it is treated as a solid wall inside the computational domain.

In the HMB2, the boundary layer correction consists in solving the flow on the grid that resolves the boundary layer in the best possible layer. All other grids in this region are non-computational, and they retrieve the flow data through interpolation. It is, therefore, possible, that a higher level grid has a hole cut by the lowest level, if the lowest level resolves the boundary layer of a solid object in the best way. In such case, the higher level grid has *interpolation holes* that receive the data from the lowest level grid.

1.6 CFD Flow Solver

The Helicopter Multi-Block(HMB2) method [5], developed at Liverpool can solve the Navier-Stokes equations in integral form using the arbitrary Lagrangian Eulerian (ALE) formulation for time-dependent domains with moving boundaries:

$$\frac{d}{dt}\int_{V(t)} \vec{\mathbf{w}} dV + \int_{\partial V(t)} \left(\vec{F_i}\left(\vec{\mathbf{w}}\right) - \vec{F_v}\left(\vec{\mathbf{w}}\right)\right) \vec{n} dS = \vec{S} \quad (4)$$

where $V(t)$ is the time dependent control volume, $\partial V(t)$ its boundary, $\vec{\mathbf{w}}$ is the vector of conserved variables $[\rho, \rho u, \rho v, \rho w, \rho E]^T$. $\vec{F_i}$ and $\vec{F_v}$ are the inviscid and viscous fluxes, including the effects of the time dependent domain. For hovering rotor simulations, the grid is fixed and a source term $\vec{S} = [0, -\rho\vec{\omega} \times \vec{u}_h, 0]^T$ is added to compensate for the inertial effects of the rotation. $\vec{u}_h$ is the local velocity field in the rotor-fixed frame of reference.

The Navier-Stokes equation are discretised using a cell-centred finite volume approach on a multi-block grid, leading to the following equations:

$$\frac{\partial}{\partial t}\left(\mathbf{w}_{i,j,k} V_{i,j,k}\right) = -\mathbf{R}_{i,j,k}\left(\mathbf{w}_{i,j,k}\right) \quad (5)$$

where $\mathbf{w}$ represents the cell variables and $\mathbf{R}$ the residuals, while i, j and k are the cell indices and $V_{i,j,k}$ is the cell volume. Osher's [6] upwind scheme is used to discretise the convective terms and MUSCL variable interpolation is used to provide third order accuracy. Van Albada limiter is used to reduce the oscillations near steep gradients.

Temporal integration is performed using an implicit dual-time step method. The linearised system is solved using the generalised conjugate gradient method with a block incomplete lower-upper (BILU) pre-conditioner [7].

Multi-block structured meshes are used for HMB2. These meshes are generated using ICEM-Hexa™of Ansys. The multi-block topology allows for an easy sharing of the computational load for parallel computing. Adding sliding meshes [8], as well as allowing for mesh overlap makes the HMB2 a very flexible solver for dealing with complex geometries. Given the existing data structure of the solver, the modifications required for the mesh overlap were restricted to a small part of the code. Within HMB2, halo cells are employed by each block, and are populated from boundary conditions, bloc-to-block data exchanges, data from sliding surfaces or data from overlapping meshes. For as long as a block has correct information on the halo cells, its solution can be updated and then shared with other neighbouring blocks. In the strongly implicit HMB2 method, only the preconditioner employed for the solution of the linear system of equations is block - de-coupled. For overlap regions though and to minimise the exchange of data, the Jacobian matrix is also de-coupled for overlapping mesh regions. Another necessary modification for the solution on overlapping grids is related to the treatment of cells marked as holes. These are identified and kept in the original system of equations even if their solution is not to be updated. This allows for the structure of the solver to remain the same and has minimal overhead in the computation.

2 Results and Discussion

2.1 2D Cases

2.1.1 NACA23012 oscillating aerofoil

The mesh sizes for an oscillating NACA23012 aerofoil case are shown in table 2. Figure 19 shows the comparison of the lift, drag and pitching moment coefficients, with matched grid results. Additional comparisons are shown for the eddy viscosity in the flowfield (see Fig. 20). The flow simulation was carried out with HMB2 at $\mathrm{M}_\infty = 0.088$ and $\mathrm{Re}_\mathrm{c} = 6.0 \cdot 10^5$ using the $k - \omega$ SST turbulence model, for both matched and overset, multi-block, structured grids. The aerofoil was pitching with the reduced harmonic frequency of a f=0.1 about a mean angle of 10 degrees and with half-amplitude of 10 degrees. The results suggest that small differences between the matched and overset grids exist at a narrow range of pitch angles during the down-stroke motion of he section. This is attributed to the better resolution of the wake flow and the shed vortices from the section, provided with the overset grid. The overset grid benefits from a regular background mesh in this case and consequently it preserves better he flow structures than the skewed mesh employed away from the blade for the fully-matched case.

2.1.2 NACA0012 aerofoil with Gurney Flap

Th mesh sizes for this case are shown in table 3. The grid was constructed using two levels, aerofoil with the mesh around it, and a Gurney flap mesh. The Gurney flap pivot point was located at a chord-wise position corresponding to $0.95c$ and the flap length was $3\%c$. The unsteady case was run with the reduced frequency of $f^* = 0.1$ with the non-dimensional time step t=0.001091, based on characteristic length travel time (*i.e* time required to travel the characteristic length with the velocity of the free stream). The comparison of C_p contours (Fig. 22 (a) and (b)) show that there is a little disturbance of pressure in the vicinity of the Gurney flap mounting point, due to the geometry change, following the boundary layer correction method. The comparison of integrated loads (Fig. 23), however, shows that there is very little effect of the geometry change on the computed lift, drag and pitching moment of the simulated aerofoil with Gurney flap configuration.

2.2 3D Cases

2.2.1 NACA0015 wing

The simplest 3D case for chimera validation was a NACA0015 wing with a rounded tip. The mesh sizes for this case are shown in table 4. The flow around the wing was computed for $\mathrm{M}_\infty = 0.5$ at $\mathrm{Re}_\mathrm{c} = 1.0 \cdot 10^6$ and incidence of $\alpha = 5°$. The grid covered half of the wing, aspect ratio of 6.6. Symmetry BC was used at the inner side of the wing. The chimera grid consisted of three levels, one for the background, one for the wing and a fine, Cartesian mesh downstream the wing tip, to resolve the tip vortex (see Fig. 24). Pressure coefficient distributions for selected wing sections are presented in Fig. 25. The distribution of Cp on the wing surface shows no difference between the matched grid case and chimera case (Fig. 25a). Cp distributions around three wing cross-sections were also plotted (Fig. 25b, c and d) and show no difference between the matched and chimera results.

2.2.2 NACA0015 wing with a fixed Gurney flap

The NACA0015 wing, from the previous chapter was equipped in a part-span Gurney flap. The grid was made with 3 levels, very coarse background grid, wing grid and Gurney flap grid. The Gurney flap was fixed at the position corresponding to -45 degrees with respect to the full actuation. The Localisation of the grids is visualised and shown in Fig. 26. The pressure coefficient distribution is shown in Fig. 27, showing the effect of the part-span Gurney flap. As predicted, more suction is generated on the suction side of the wing in the span-wise region, where the Flap is mounted on the pressure side. At the same time, the Gurney flap increases the pressure on the lower side of the wing.

2.2.3 ONERA 7AD rotor in hover

The next case, was a rotor in hover. Mesh sizes for this case are shown in table 5. The ONERA 7AD rotor blade grid was built of one layer of blocks around it. The background grid was radial, to allow for periodic hover boundary conditions. The blade was computed at a collective of 7 degrees and at tip Mach number of 0.6. The flow was assumed to be periodic in space and time and so only a single blade of this 4-bladed rotor was used. The mesh topology and density follow previous experience with HMB2 [5]. The overset mesh flagging (shown in Fig. 28) and localisation was performed in a very efficient fashion. The solution captured well the blade loading that is compared in Figure 28(a) with a result obtained using a matched grid. For a good comparison, three blade slices were extracted and the values of pressure coefficient Cp are compared with a matched grid solution.

For further comparison, the pressure coefficient was plotted of 3 different sections of the blade ($r/R = 0.7$, $r/R = 0.825$, $r/R = 0.915$). The matched grid and chimera results show very good agreement with experimental data.

2.2.4 UH60A, rotor in forward flight

The mesh sizes for UH60A rotor in forward flight case are shown in table 6. This case has been computed with $\kappa - \omega$ turbulence model at high advance ratio ($\mu = 0.368$, collective $\theta_0 = 11.6°$). Additionally, prescribed aeroelastic blade twist was applied, with 5 harmonics as in [8]. Three levels were used for the computation: background, disc and rotor. The disc was used as an intermediate level, between the relatively coarse background grid and fine grids of rotor blades (see Fig. 30).

During the chimera computations all blocks associated with rotor were treated as rigid. Principally, when the chimera method is used for rotorcraft CFD, each blade is covered with a thin layer of blocks ($\sim 50\% - 100\%$ of a cord length). As a blade pitches, the grid associated with it is rigidly rotated / translated. This allows for maintaining the grid quality at the trailing edge throughout the rotor revolution.

The comparison of lift around the azimuth between the matched and overset grid results(see Fig. 31) shows very good agreement of the results, especially in the advancing side. The prediction of shock-wave effect on the advancing side is in almost perfect agreement between two methods. Very small differences can be seen at the inboard section of the retreating side. Additionally, the distribution of lift around the azimuth at constant radius was compared between the matched and overset grid and the experimental results (see Fig. 32). Numerical results for two different methods follow each other almost perfectly, whereas some solution inaccuracy can be seen compared to experimental results.

At the end, the wake visualisation behind the rotor was shown (Fig. 33). The wake is preserved very well due to the fact that it is propagated inside the background grid, which in this case is artesian, and provides low dissipational properties.

2.2.5 ROBIN case

The ROtor Body INteraction case (ROBIN) was simulated using the Euler method of HMB2. Mesh sizes for this case are shown in table 7. The advance ratio was set to $\mu = 0.15$, collective $\theta_0 = 6.5°$, as in [5], the grid consists of four levels: regular, Cartesian background grid, body fitted fuselage grid, disc grid and rotor grid (see Fig. 34). This configuration allows for the regular cell shapes in the background grid, for good resolution of vortical structures in the wake. A disc grid is needed as an intermediate level, between fine blade grids and relatively coarse background grid in the rotor region.

For the comparison of the results with the fuselage and the rotor, The plots for thrust coefficient, torque coefficient and moment coefficient were created (see Fig. 35). The differences between the torque, thrust and moments are minor. The interaction between the rotor and fuselage was visualised by the extraction of unsteady pressure on the fuselage, at different azimuth angles. Figure 36 shows the differences in the distribution of unsteady pressure for the azimuth angles of $220°$, $240°$, $260°$ and $360°$, between the chimera case and sliding plane case. Although unsteady pressure calculation is hard to capture, and in both sliding plane method and overset mesh method dissipation exists at the interface between the grids, the unsteady pressure plots exhibit good agreement for various azimuth angles.

Figure 37 shows the wake visualisation behind the ROBIN with use of Q criterion iso-surfaces. The wake is preserved very well, both in the proximity of the rotor and downstream from it. The sudden brake-down of vortical structures is caused by the coarse mesh, where the structure of the vortices cannot be resolved.

3 Conclusions

In this paper an overset mesh method was presented and demonstrated. A hierarchical approach was used that is compatible with the HMB2 CFD solver and resulted in fast turn-around times for the localisation of overset grids on relatively complex cases. The method has enough flexibility to allow for arbitrary mesh overlaps and is designed with parallel execution in mind. The key ingredients of the method are the use of a tree-structure to represent the mesh and facilitate the localisation and an efficient local search for cell overlap and an efficient, combined method for a *cell-in-solid* localisation, that relies on the MVBB pre-search and ray tracing search. Although the chimera method presented in this paper is based on a non-conservative interpolation algorithm, the effects of non conservativeness have been mitigated. The comparison of chimera method with sliding mesh and matched grids shows that the agreement between the results is satisfactory. Additionally, extensions to the chimera method of HMB2 were described. The implementation of ray tracing made the detection of cells in solid surface possible for very complex shapes, even if the grid associated with them consists of one thin layer of blocks. The Boundary Layer Correction method is a step forward towards the computations of closely coupled geometries, like Gurney flaps. It has been shown that, despite the fact the solution localisation in case of embedded solids leads to slight changes of geometry, the solution is not substantially influenced.

Acknowledgements
This work was carried out as part of Rotor Technology Validation Programme (RTVP) of the UK Technology Strategy Board and AgustaWestland Ltd.

References

[1] M. Jarkowski, M. Woodgate, J. Rokicki, and G.N. Barakos. Towards hybrid consistent overset mesh method for rotorcraft cfd.

[2] M. Woodgate and G.N. Barakos. Impicit CFD Methods for Fast Analysis of Rotor Flows. 36th European Rotorcraft Forum, Paris, France, September 2010. .

[3] J.R. Shewchuk. Robust Adaptive Floating-Point Geomeric Predicates. pages 141–150. ACM, Proceedings of the 12th Annual Symposium on Computational Geometry, May 1996. .

[4] J.R. Shewchuk. Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. *Discrete & Computational Geometry*, 18(3):305–363, 1997.

[5] R. Steijl, G. N. Barakos, and K. Badcock. A framework for CFD analysis of helicopter rotors in hover and forward flight. *International Journal for Numerical Methods in Fluids*, 51(8):819–847, 2006.

[6] S. Osher and S. Chakravarthy. Upwind Schemes and Boundary Conditions with Applications to Euler Equations in General Geometries. *Journal of Computational Physics*, 50(3):447–481, June 1983.

[7] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press: Cambridge, MA, 1994.

[8] R. Steijl and G. N. Barakos. Sliding Mesh Algorithm for CFD Analysis of Helicopter Rotor-Fuselage Aerodynamics. *Int. J. Numer. Meth. Fluids*, 58:527–549, 2008.

Table 1: Sample CPU times on 4 cores of Intel Xeon CPUs for the Overset Mesh functionality in HMB.

Test Case	Total mesh size (blocks per level)	Localisation time [s]
Aerofoil NACA0012	18,104 (12,8)	below 1
Wing NACA0015	3,999,744 (60,20)	12
7AD rotor in hover	2,942,892 (112,48)	12
UH60 Forward Flight	11,423,440 (210, 240, 160)	51
ROBIN Forward Flight	9,302,432 (252, 128, 48, 80)	44

Table 2: Grid size per level and number of blocks per level for NACA23012 case.

NACA23012	
No. of Levels	2
Mesh Size Total / No.Blocks Total	111,742 / 41
Mesh Size Level 0 / No.Blocks in Level 0	69,630 / 25
Mesh Size Level 1 / No.Blocks in Level 1	42,112 / 16

Table 3: Grid sizes per level and number of blocks per level for NACA0012 aerofoil case

NACA0012	
No. Levels	2
Mesh Size Total / No.Blocks Total	78024 / 31
Mesh Size Level 0 / No.Blocks Level 0	56840 / 24
Mesh Size Level 1 / No.Blocks Level 1	21184 / 7

Table 4: Grid size per level and number of blocks per level for a triple-overlap wing, NACA0015 case.

NACA0015	
No. of Levels	3
Mesh Size Total / No.Blocks Total	4,136,924 / 100
Mesh Size Level 0 / No.Blocks in Level 0	2,488,320 / 60
Mesh Size Level 1 / No.Blocks in Level 1	1,511,424 / 20
Mesh Size Level 2 / No.Blocks in Level 2	137,180 / 20

Table 5: Grid size per level and number of blocks per level for hovering rotor, 7AD case.

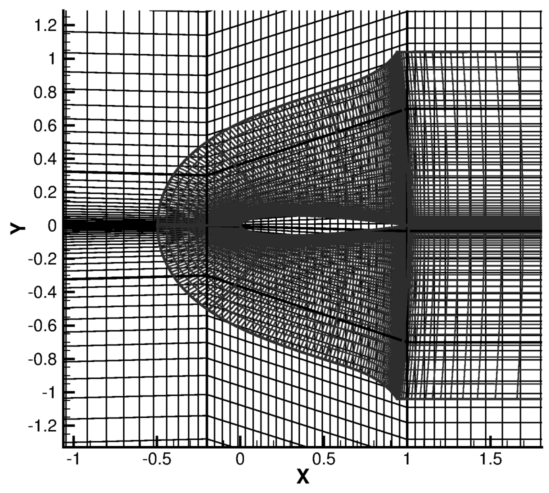
7AD blade	
No. of Levels	2
Mesh Size Total / No.Blocks Total	2,942,892/ 150
Mesh Size Level 0 / No.Blocks in Level 0	2,592,000 / 112
Mesh Size Level 1 / No.Blocks in Level 1	350,892 / 48

Table 6: Grid size per level and number of blocks per level for forward-flying rotor, UH60 case.

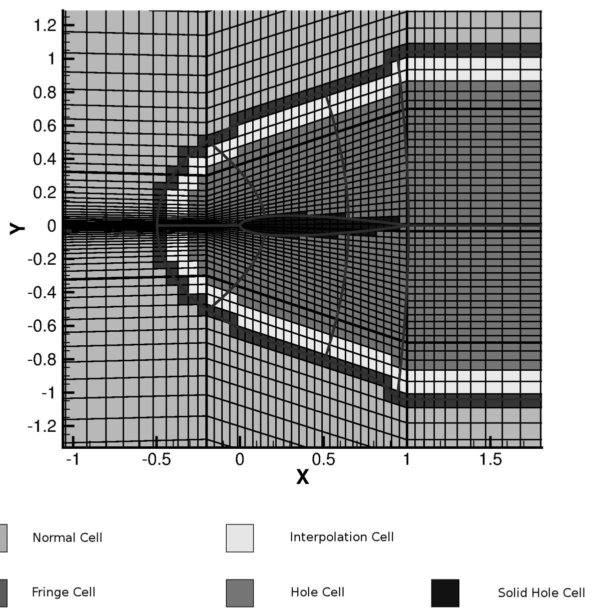
UH60	
No. of Levels	3
Mesh Size Total / No.Blocks Total	12,756,822 / 610
Mesh Size Level 0 / No.Blocks in Level 0	1,957,926 / 210
Mesh Size Level 1 / No.Blocks in Level 1	4,644,528 / 240
Mesh Size Level 2 / No.Blocks in Level 2	6,154,368 / 160

Table 7: Grid size per level and number of blocks per level for ROBIN case.

ROBIN	
No. of Levels	4
Mesh Size Total / No.Blocks Total	10,391,606 / 508
Mesh Size Level 0 / No.Blocks in Level 0	4,736,732 / 252
Mesh Size Level 1 / No.Blocks in Level 1	2,084,000 / 128
Mesh Size Level 2 / No.Blocks in Level 2	886,320 / 48
Mesh Size Level 3 / No.Blocks in Level 3	3,082,544 / 80

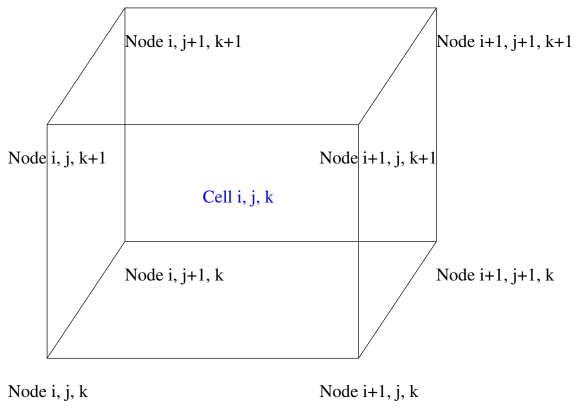


(a)



(b)

Figure 1: Overset grids covering the domain around an aerofoil.



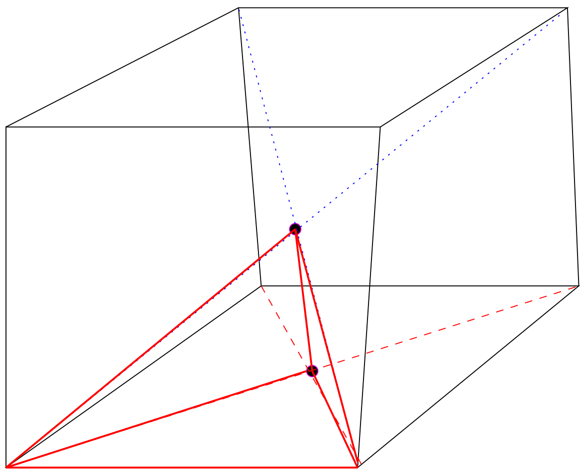


Figure 2: (a) Cell and associated nodes. All cells are unambiguously referred to by giving the lowest i, j and k indices of the nodes. (b) Hexahedron split into 24 tetrahedra. Each cell face is split into four triangles with a common apex. Each tetrahedron is thus split in 24 tetrahedra, with a common apex located at the cell centroid.

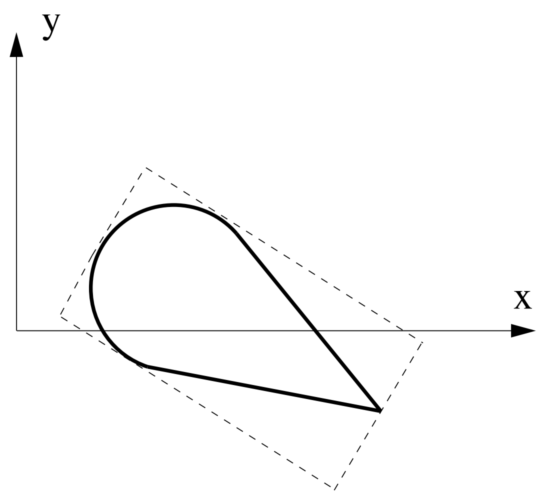


Figure 3: Minimum Volume Bounding Box (MVBB) around a 2D shape. The thick lines represent the block, the dashed line its Minimum Volume Bounding Box.

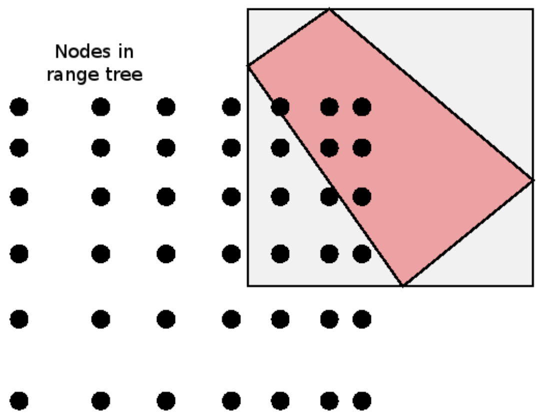


Figure 4: Range-tree schematic. Points represent all nodes in a level. To find the nodes located inside the trapezoidal cell (belonging to another level) at the top right side, the tree has to be queried for the points within the cell's bounding box. Next, the positions of the nodes returned by the tree are checked with respect to the cell.

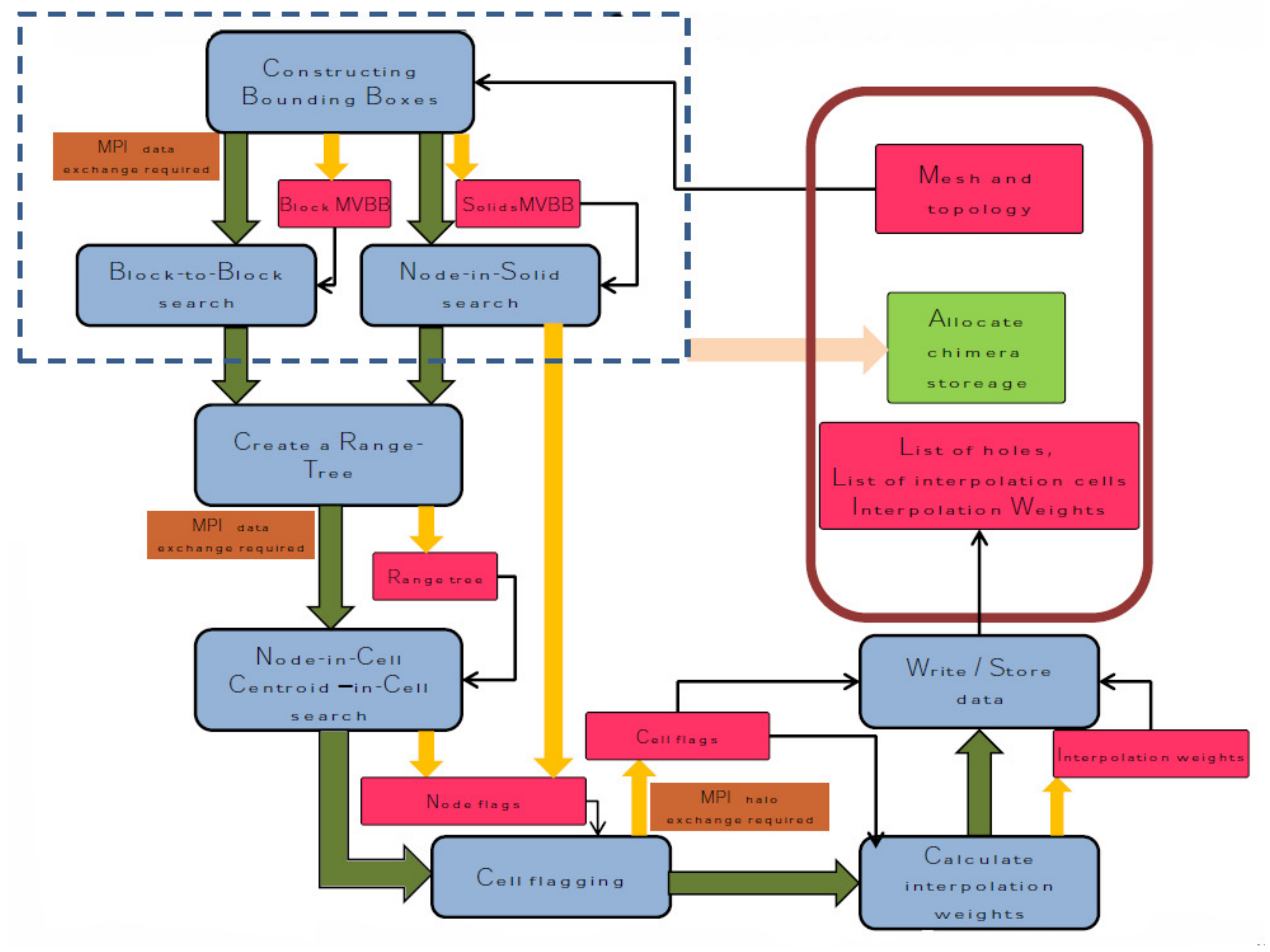

Constructing Bounding Boxes
MPI data exchange required
Block MVBB
Solids MVBB
Block-to-Block search
Node-in-Solid search
Mesh and topology
Allocate chimera storeage
List of holes, List of interpolation cells Interpolation Weights
Create a Range-Tree
MPI data exchange required
Range tree
Node-in-Cell Centroid -in-Cell search
Write / Store data
Cell flags
Interpolation weights
Node flags
MPI halo exchange required
Cell flagging
Calculate interpolation weights

Figure 5: Schematic of the chimera Method in HMB. Rounded boxes represent set of operations, dashed are data sets. Arrows refer to the order of operations, dashed arrows represent data on the output, thin arrows represent data input. Dashed box (top left) refers to LP. The chimera method procedure uses mesh and topology data, which is stored in the CFD part of the solver. After a set of operation, called Localisation Pre-processing, the storage for chimera is allocated.

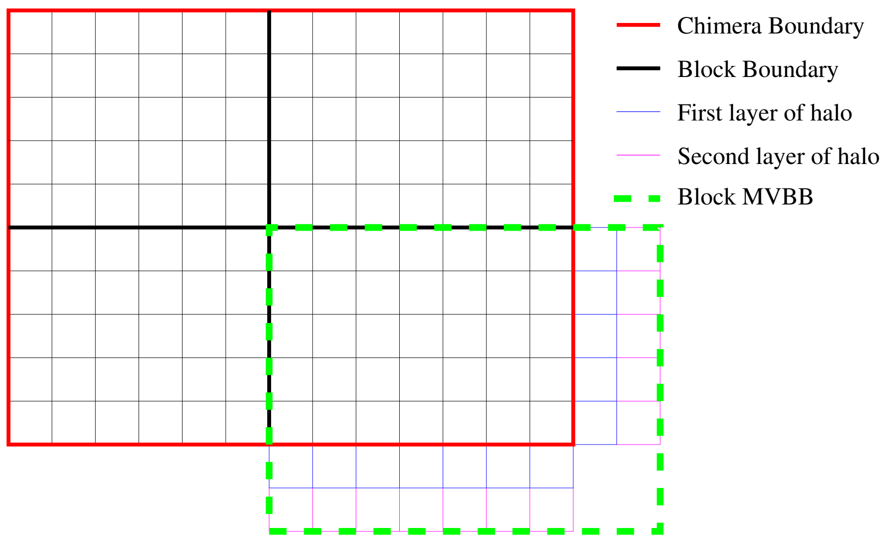
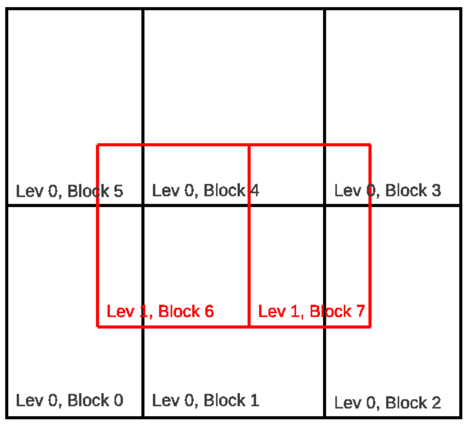


Figure 6: MVBB shown for one of four blocks in a chimera grid. Two layers of halo cells are added on a chimera boundary, and are included in the MVBB. Halo mesh of all other types of boundaries is ignored in the MVBB generation.



Host: 0, Neighbours: 6;
Host: 1, Neighbours: 6, 7;
Host: 2, Neighbours: 7;
Host: 3, Neighbours: 7;
Host: 4, Neighbours: 6, 7;
Host: 5, Neighbours: 6;
Host: 6, Neighbours: 0, 1, 4, 5;
Host: 7, Neighbours: 1, 2, 3, 4

Figure 7: (a) Multi-block topology with two different levels in 2D. In this case level 0 has 6 blocks (0 to 5) and level 1 has 2 blocks (6 and 7). (b) The groups are created for load_balancing:
The number of hosts is the same as number of blocks and some of the blocks appear as neighbours more than once (*ex.* block 7 is a neighbour of groups 1, 2, 3 and 4).

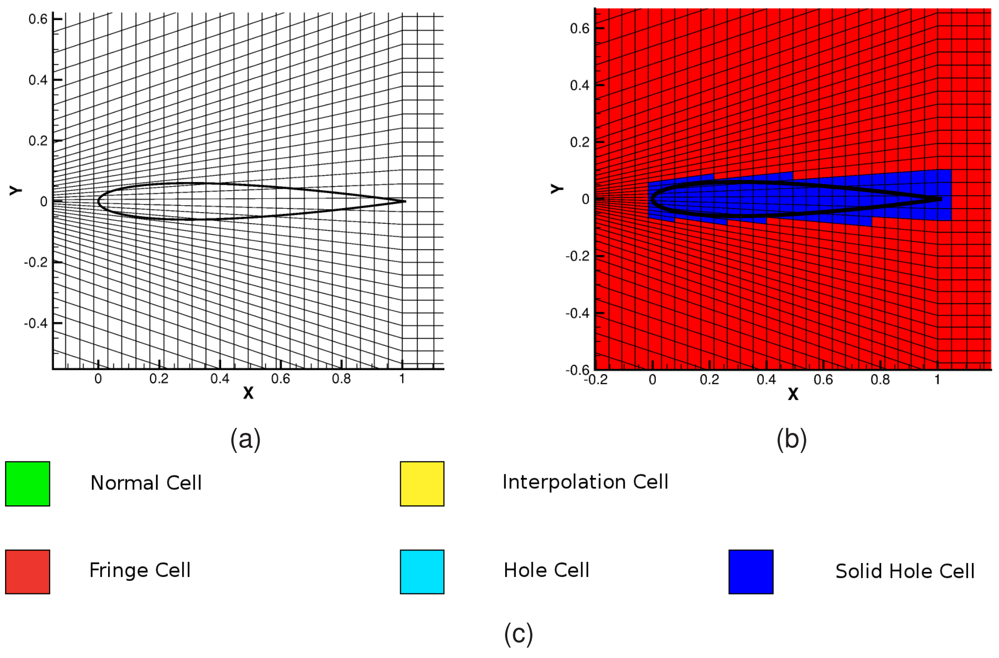


Figure 8: (a) Aerofoil shape within a background grid. All the nodes inside the aerofoil have to be identified and flagged as nodes-in-solid. In the first step, a minimum volume bounding box of the solid shape is created. (b) All the nodes within solid MVBB have been identified. If at least one vertex of a cell is localised inside the solid MVBB, the cell is presented in dark. It has to be noted, however, that at this point the flags refer to vertices rather than cells. It has to be noted that the solid pattern is different here than in Fig. 1b. The reason for this is that the flags shown here refer to nodes in solid MVBB rather than cells bounded by solid surface. The operations that lead to the final set of flags are explained below.

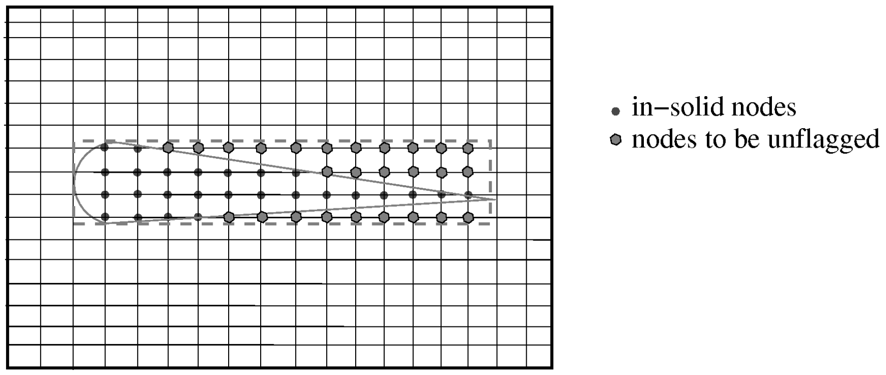


Figure 9: Node unflagging procedure. Dashed line represents the MVBB of a solid shape (aerofoil). All the nodes inside the solid MVBB get flagged as nodes in solid. At the later stage, the nodes which are inside the solid MVBB, but outside of the solid shape, will become unflagged.

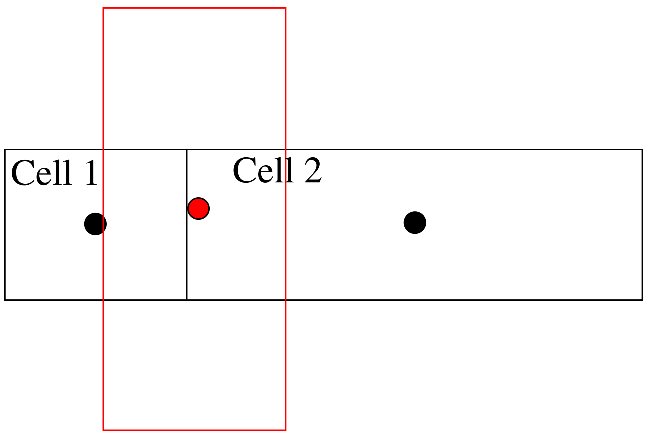


Figure 10: The centroid of the vertically aligned cell (dashed line) is located within the grid constructed of cells shown with solid line. The nearest centroid for the red point is located in cell 1, but cell 2 is the one, by which the centroid is bounded.

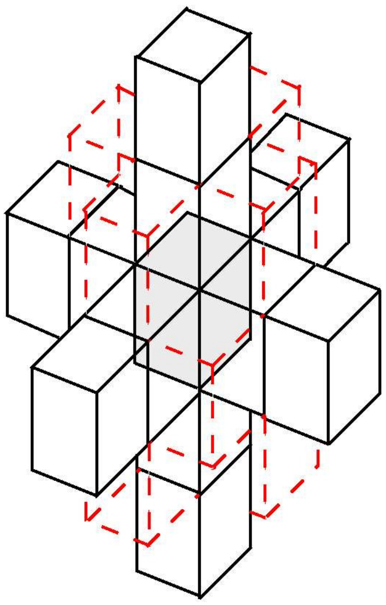
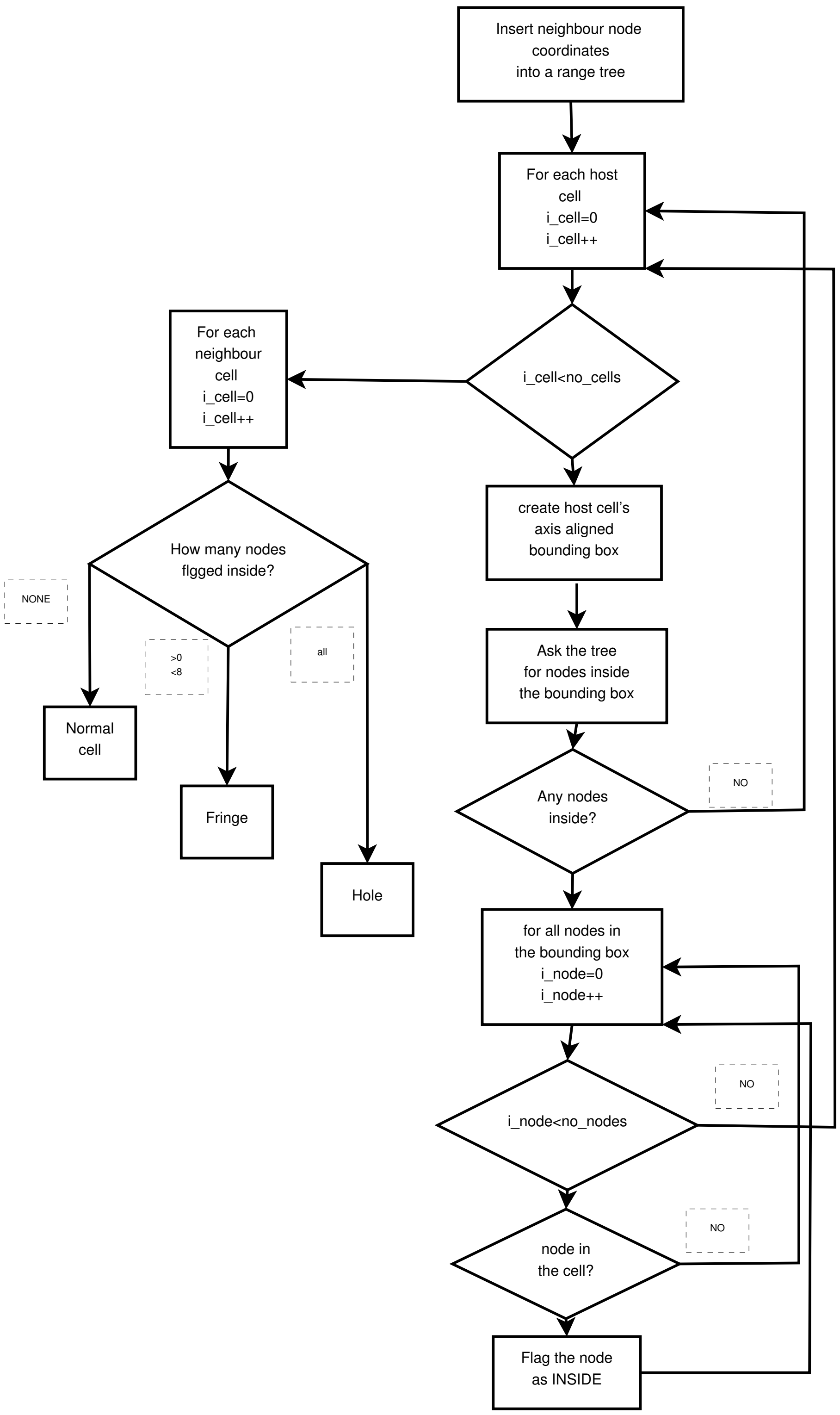


Figure 11: Viscous stencil for MUSCLE in HMB.



Insert neighbour node coordinates into a range tree
For each host cell i_cell=0 i_cell++
i_cell<no_cells
For each neighbour cell i_cell=0 i_cell++
How many nodes flgged inside?
NONE
>0 <8
all
Normal cell
Fringe
Hole
create host cell's axis aligned bounding box
Ask the tree for nodes inside the bounding box
Any nodes inside?
NO
for all nodes in the bounding box i_node=0 i_node++
NO
i_node<no_nodes
NO
node in the cell?
Flag the node as INSIDE

Figure 12: Schematic of the node flagging procedure.

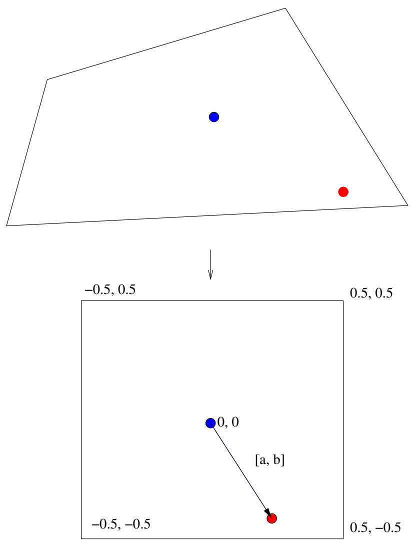


Figure 13: For second order interpolation with flow variable distribution reconstruction, the cell containing the interpolation cell centroid has to be normalised and represented in a Cartesian Coordinate System (see Fig. 2b) Here, the operation is presented in 2D. The normalised cell has a unitary edge length and its centroid is located in $[0, 0]$. Any point inside the normalised cell can be represented by a distance vector $[a, b]$, where a and b are within the range $[-0.5, 0.5]$.

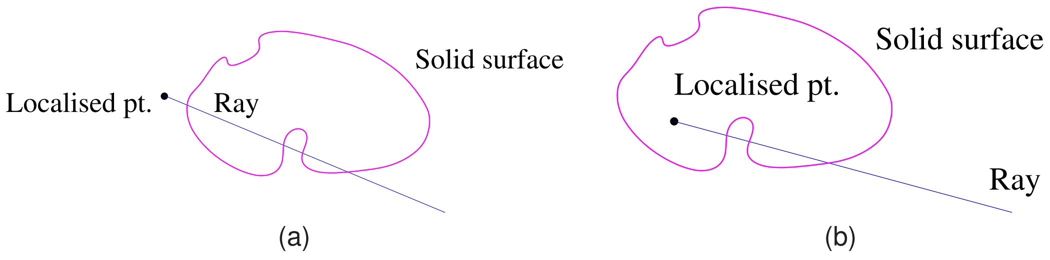


Figure 14: Schematic of a ray-tracing algorithm. (a) An even number of ray-solid surface intersections indicates that the point is outside of the solid, (b) an odd number of ray-solid surface intersections indicates that the point is inside of the solid.

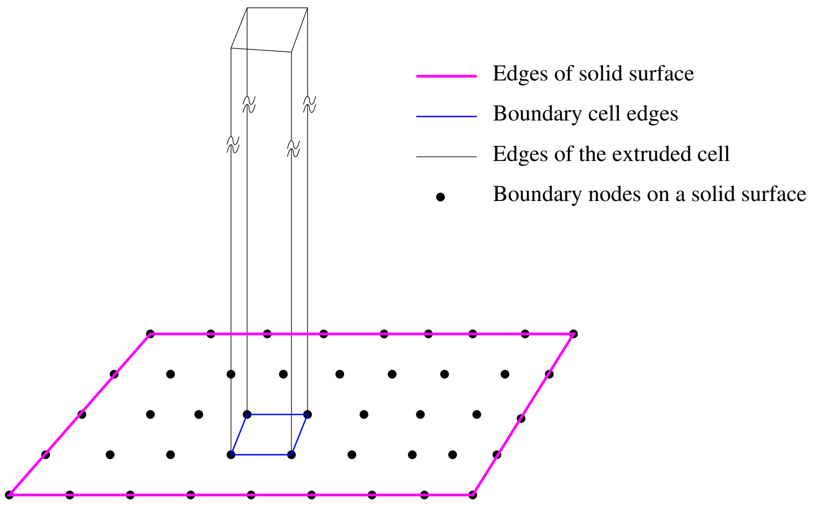
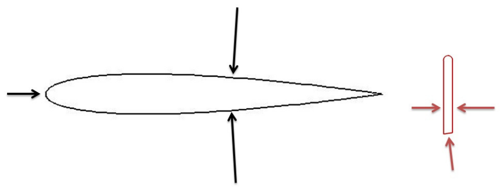
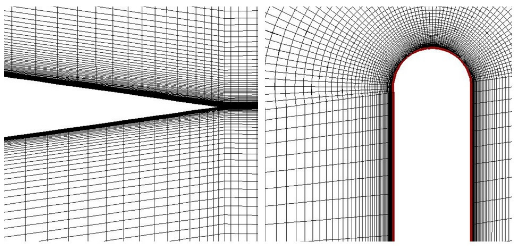


Figure 15: Semi-infinite extrusion of a boundary cell, for the Ray tracing. Boundary cell, constructed of 4 nodes is extruded along positive direction of the Z axis.

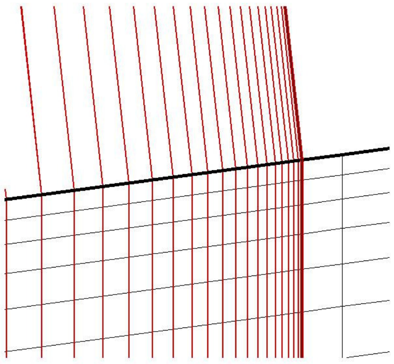


(a)

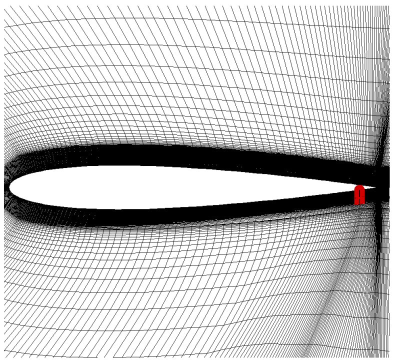


(b)

Figure 16: (a) Refinement direction on solid geometry features (b) Grid in the proximity of the aerofoil and Gurney flap. The solid features are presented in unnatural scale, so that the grid can be clearly visible.



(a)



(b)

Figure 17: (a) Overset mesh of the NACA0012 aerofoil and the Gurney flap (b) The close-up to the intersection corner between the aerofoil mesh and Gurney flap mesh. It can be seen, that the cells directly on the solid boundaries intersect each other in an unmatching manner.

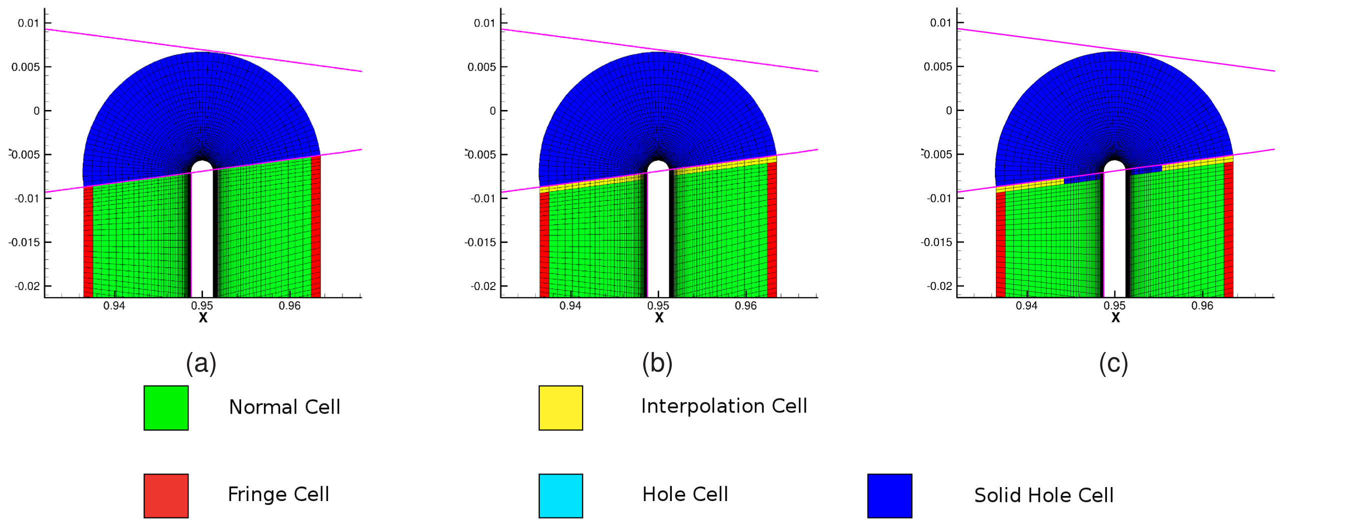


Figure 18: The step-by-step procedure of cell flagging in the boundary layer intersection case. The Gurney flap grid can be seen and the set of flags associated with it. The Gurney flap is mounted below the aerofoil grid, whose edges are presented in violet. (a) Flags after *point-in-solid* search, (b) a set of interpolation cells is appended between computational cells and *solid* cells (c) interpolation cells without a valid interpolee are flagged as *solid* cells and *computational cell* - *solid cell* pattern is obtained.

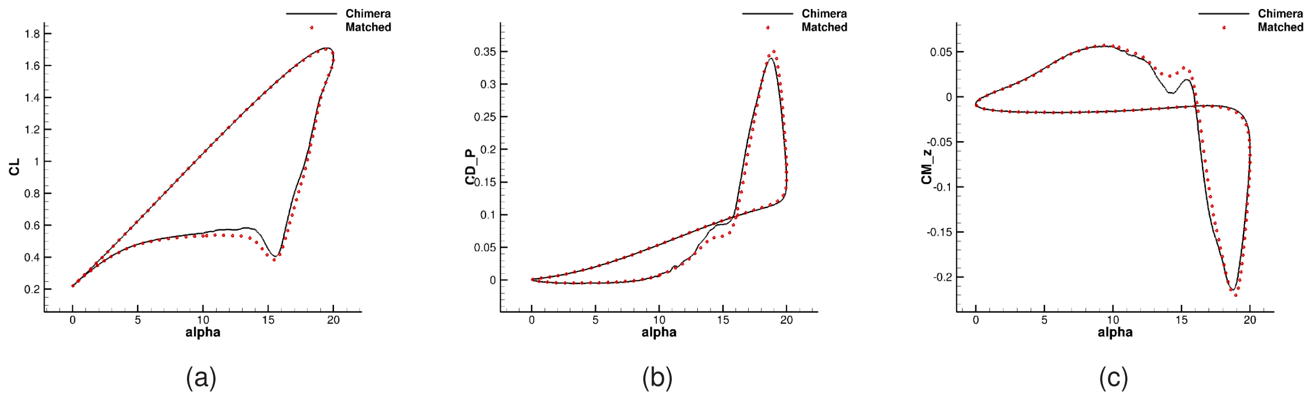
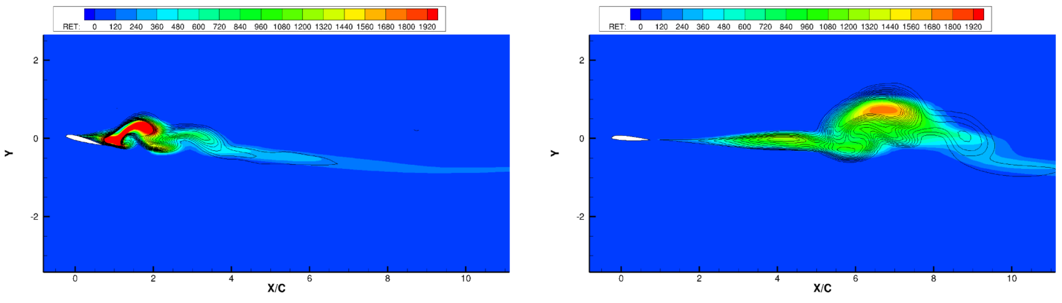


Figure 19: NACA23012 pitching aerofoil. Comparisons between chimera and matched grid solution for (a) lift coefficients against angle, (b) drag coefficient and (c) pitching moment coefficient. The cases were computed at the following conditions $\mathrm{M} = 0.088$ and $\mathrm{Re} = 6.0 \cdot 10^5$, mesh size $\sim 110,000$ for both cases.



(a) (b)

Figure 20: NACA23012 pitching aerofoil. The con ours (matched grid) and iso-lines (chimera grid) of eddy viscosity, for two different time instances of the pitching aerofoil.

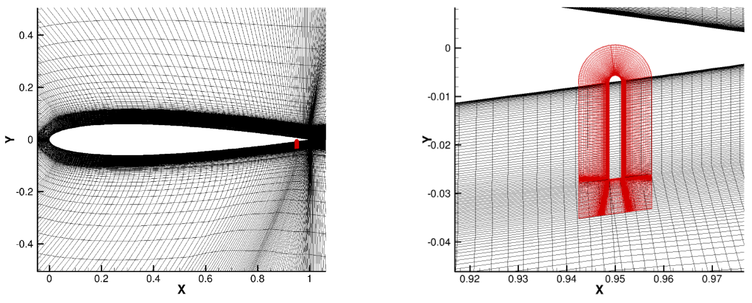
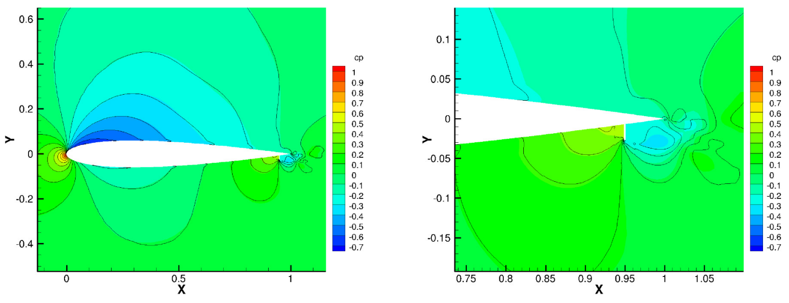


Figure 21: Grid used for the Gurney flap simulation on NACA0012 aerofoil


(a) (b)

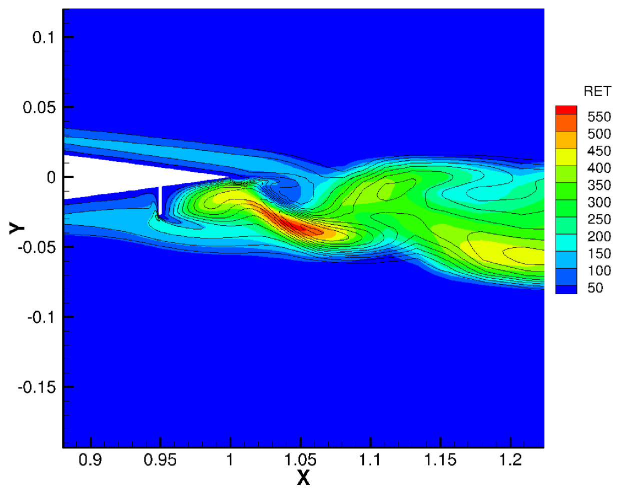

(c)

Figure 22: Comparison between the contours of pressure coefficient (a) and (b) and eddy viscosity (c) for chimera (solid line) and matched (colours) grid solution.

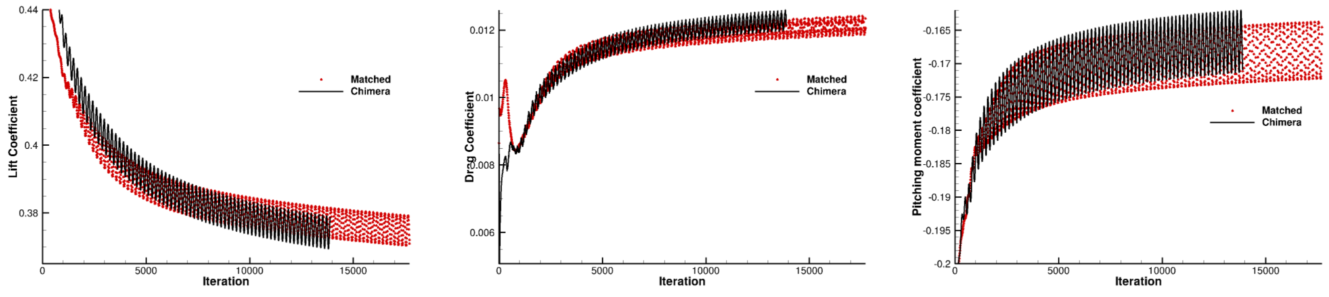

(a) (b) (c)

Figure 23: Comparison between pressure, drag and pitching moment coefficients against iteration number for a NACA0012 aerofoil with a Gurney flap.

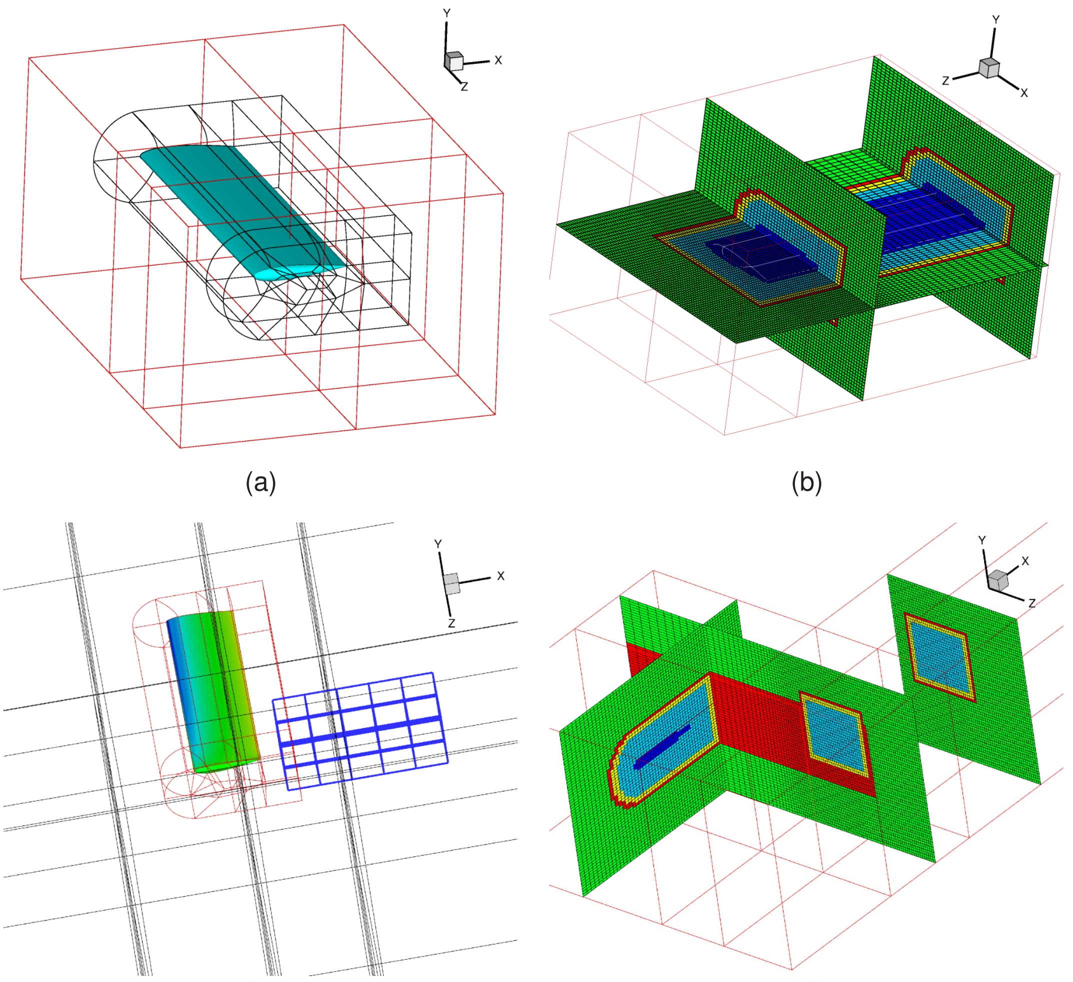
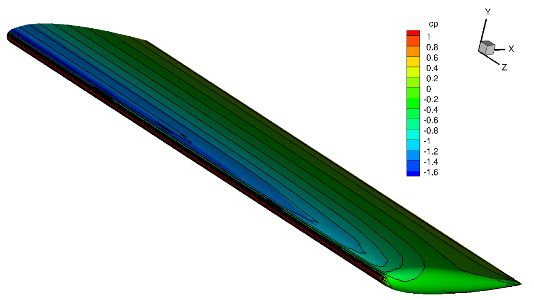
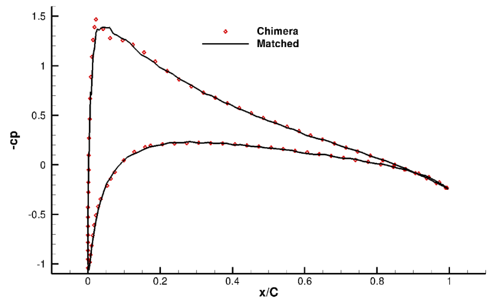


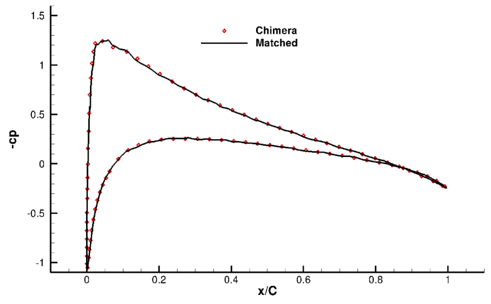
Figure 24: NACA0015 wing with double and triple overlap. Tip vortex was successfully transferred through the *chimera interface* into the fine Cartesian grid. Flow field is resolved on the overset mesh and significant flow features , *e.g.* the tip vortex were successfully captured.



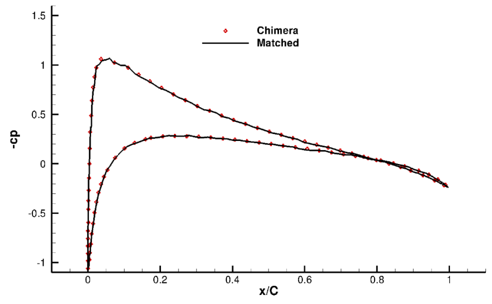
(a)



(b)

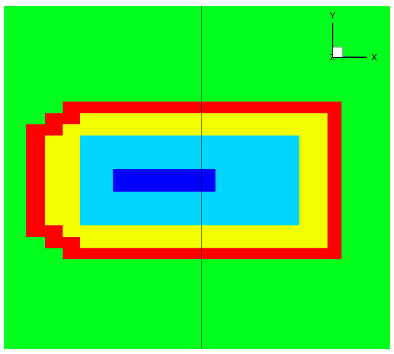


(c)

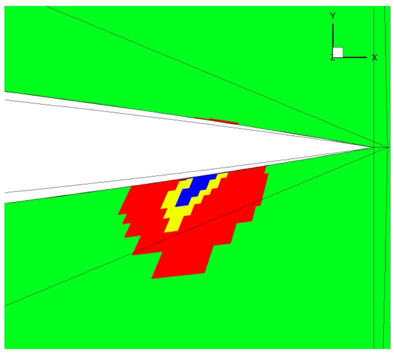


(d)

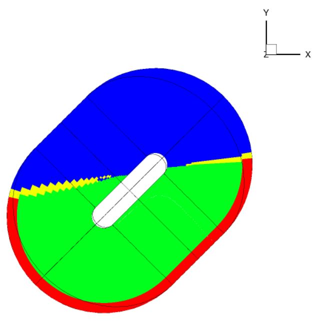
Figure 25: NACA0015 wing, (a) shows the Cp distribution on the wing's surface. Contours represent the chimera results, whereas the black iso-lines - matched case. (b), (c) and (d) show the Cp distributions at three sections: 0.5, 0.75



(a)

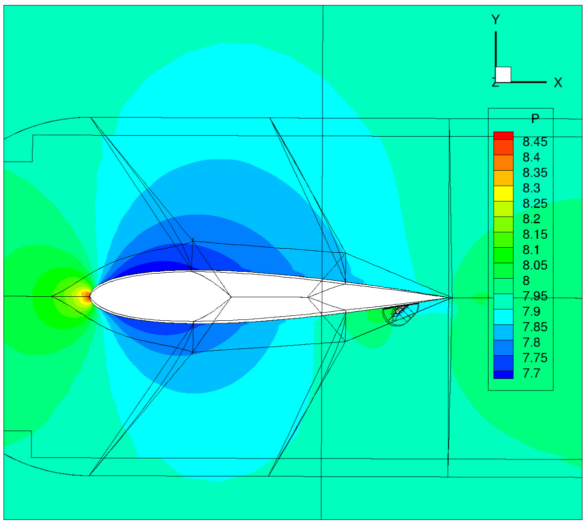


(b)

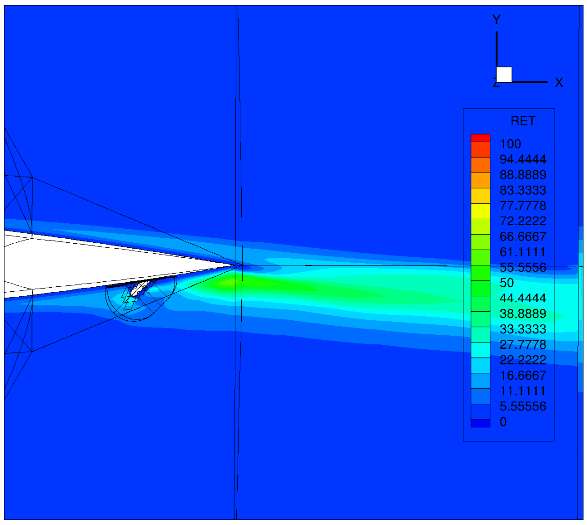


(c)

Figure 26: Localisation slices of (a) background, (b) wing and (c) Gurney flap grids in the XY cross-section

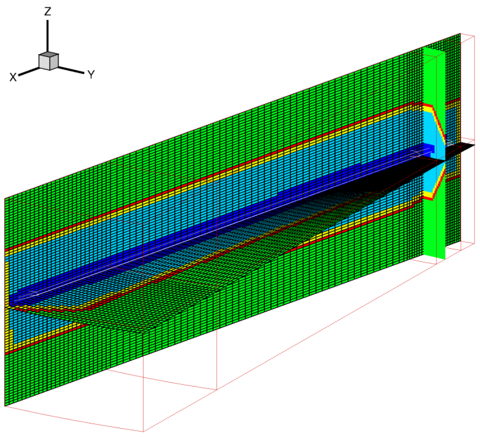


(a)

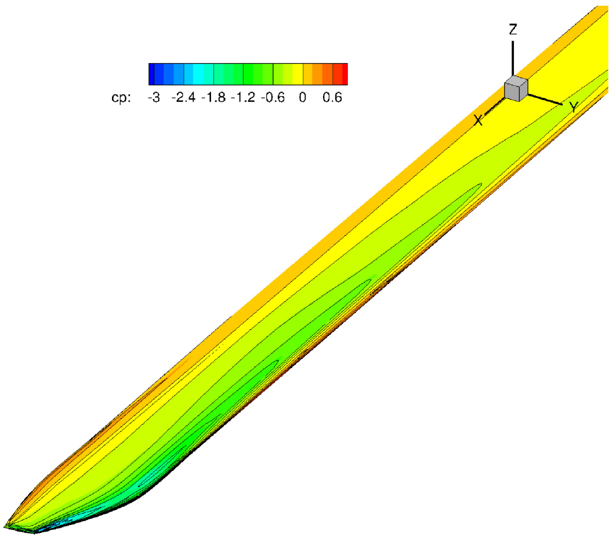


(b)

Figure 27: Slice of (a) pressure and (b) eddy viscosity in the cross section cutting through the gurney flap, located at -45 deg.

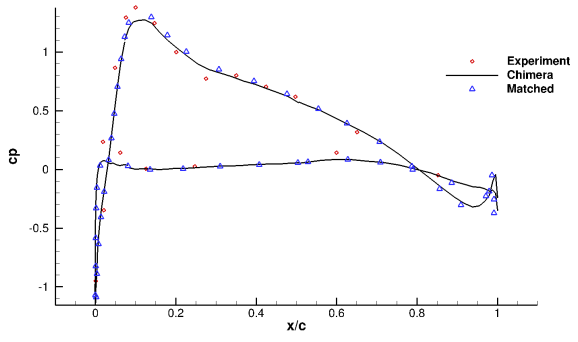


(a)

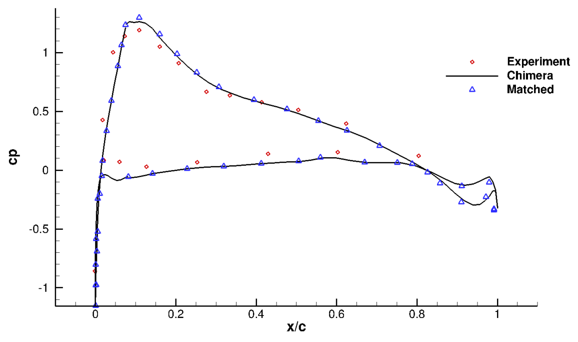


(b)

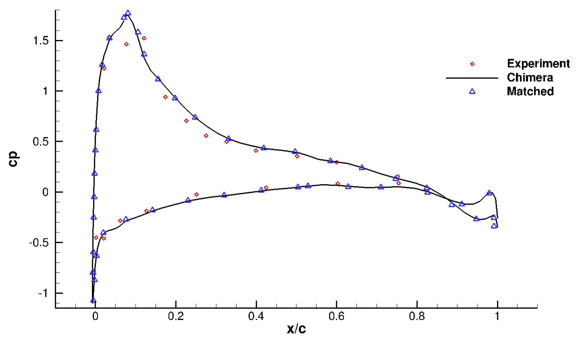
Figure 28: ONERA 7AD rotor in hover. (a) Overset mesh localisation and sample results for the case of a hovering ONERA 7AD blade. The inviscid model was used for this computation and the near-blade grid was restricted to a single layer of blocks around the blade. (b) Surface cp comparison. Solid lines correspond to the chimera solution and shaded contours to the solution on a perfectly matching mesh.



(a)

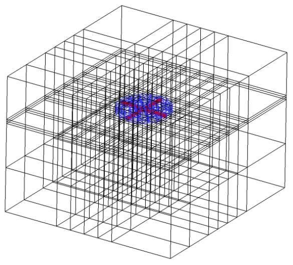


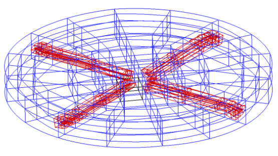
(b)

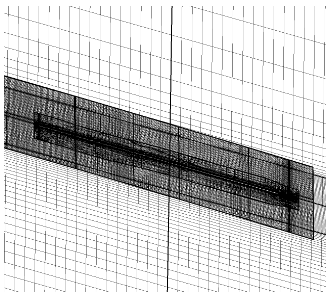


(c)

Figure 29: ONERA 7AD rotor in hover: cp comparisons between the experimental data, matched and chimera CFD case for three sections: (a) $r/R = 0.7$ (b) $r/R = 0.825$, (c) $r/R = 0.915$.


(a)


(b)


(c)

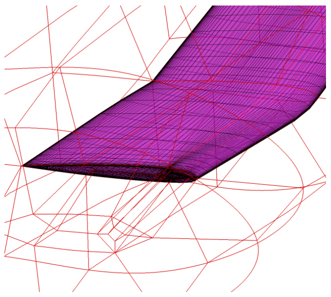

(d)

Figure 30: UH60 Rotor in Forward Flight. Topology and grid used for UH60 forward flight calculations

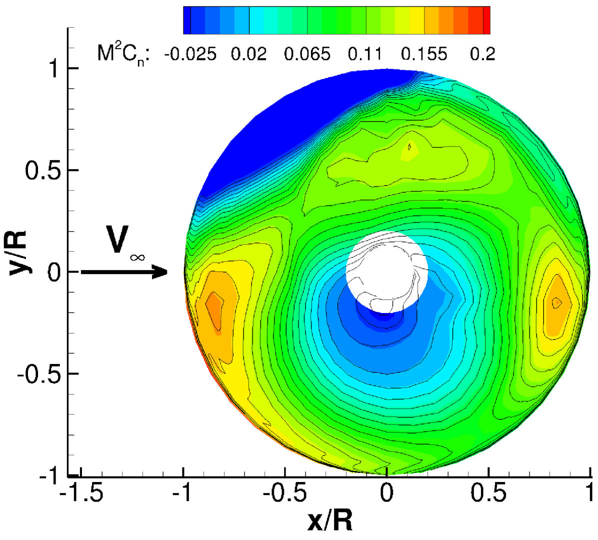
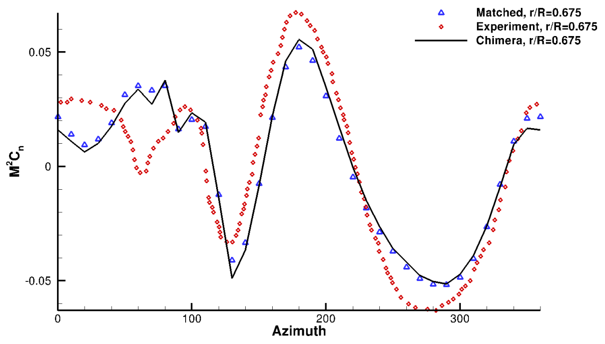
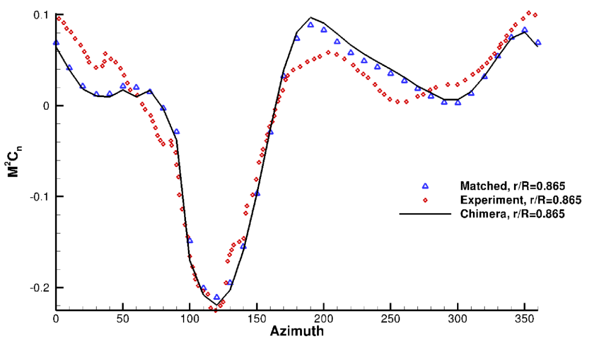


Figure 31: UH60 Rotor in Forward Flight. ^M2Cn comparison with matched grid. Contours represent the matched grid results, whereas black iso-lines are the chimera solution



(a)



(b)

Figure 32: M^2C_n comparisons between chimera results, matched grid results and experimental data at two different radial sections (a) r/R=0.675 and (b) r/R=0.865. From both experiment and CFD, the average values have been subtracted.

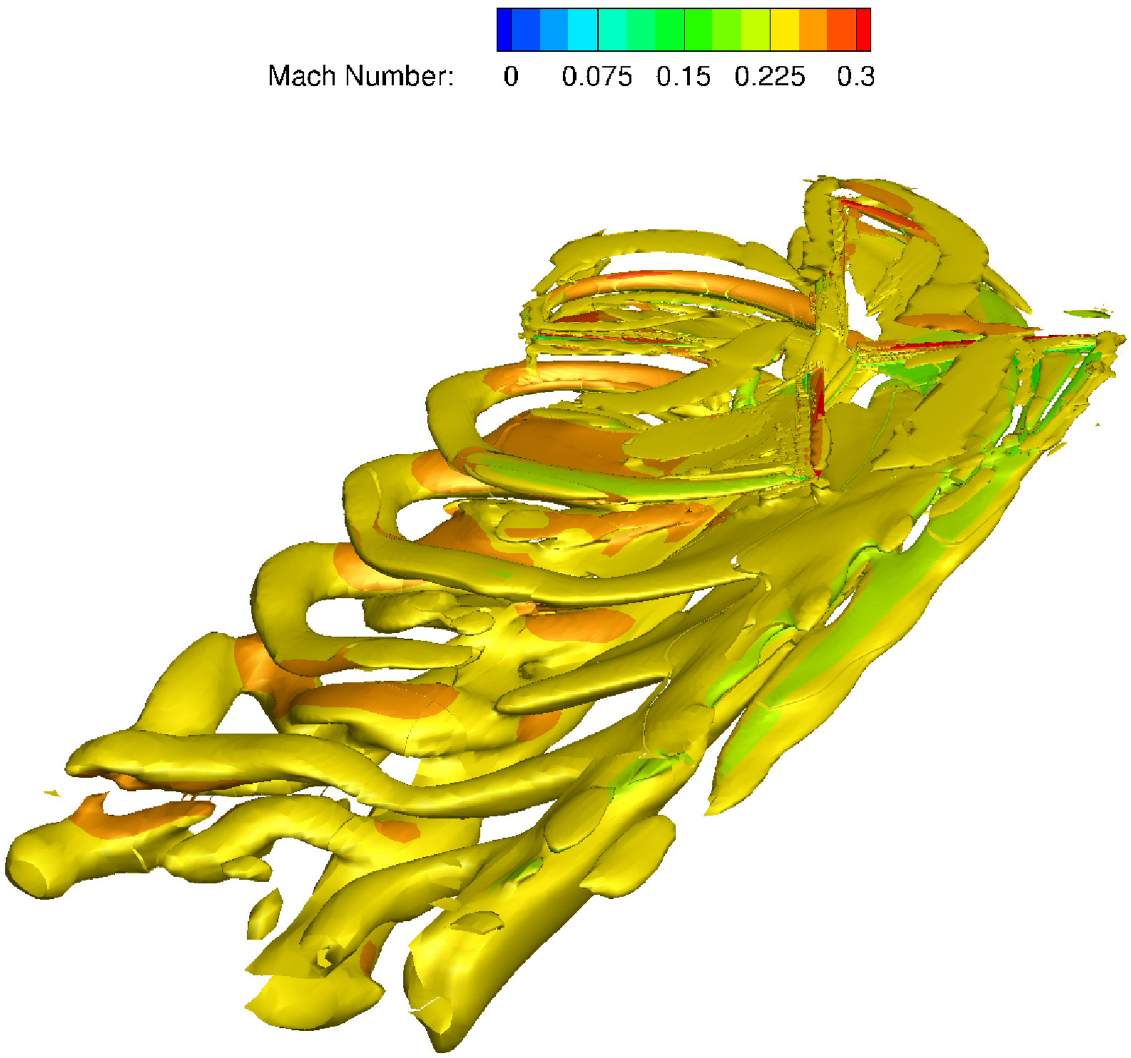


Figure 33: UH60, Forward Flight: Wake visualisation. Iso-surfaces of Q criterion are shown ($Q - 0.001$), and coloured with Mach number

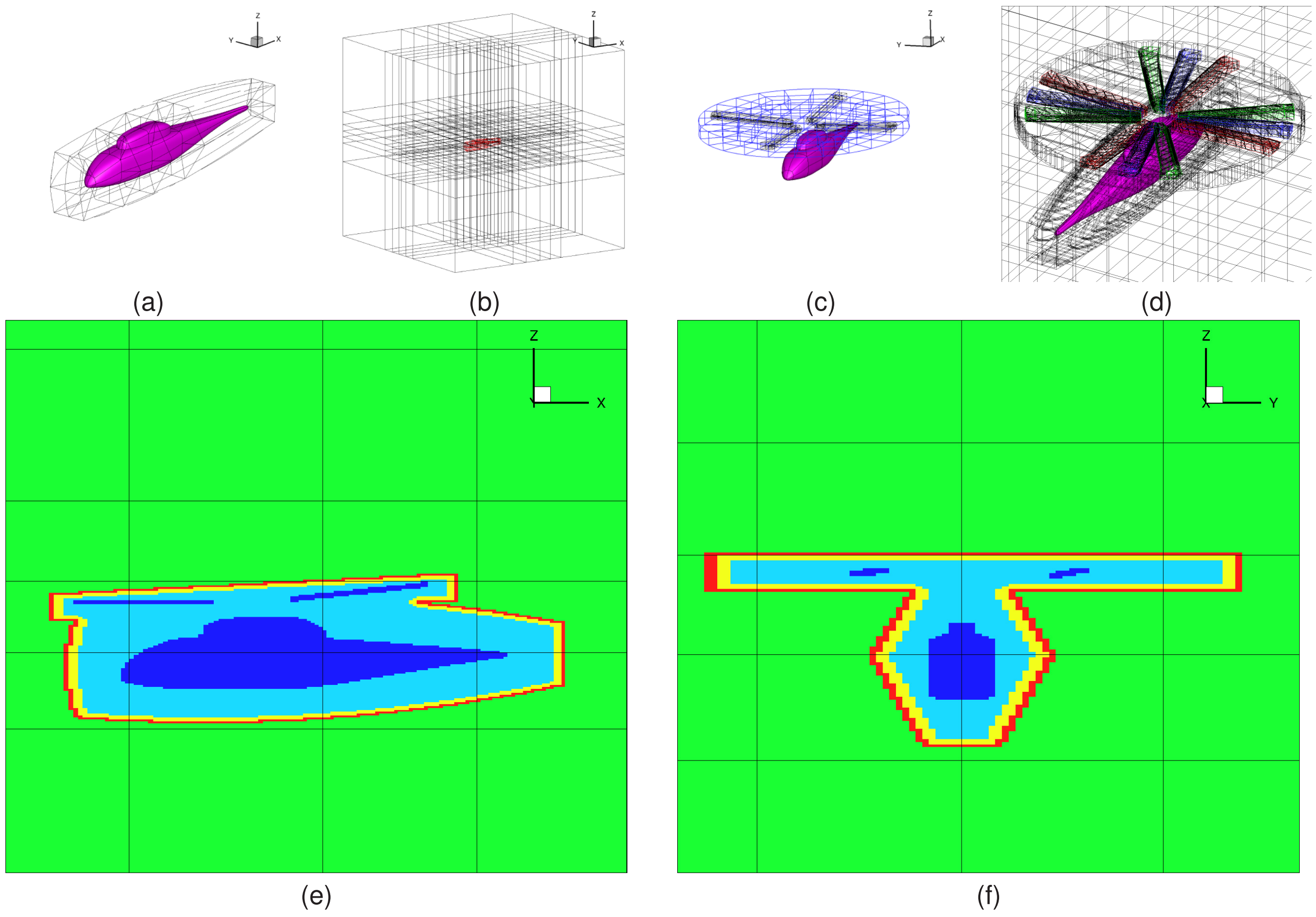
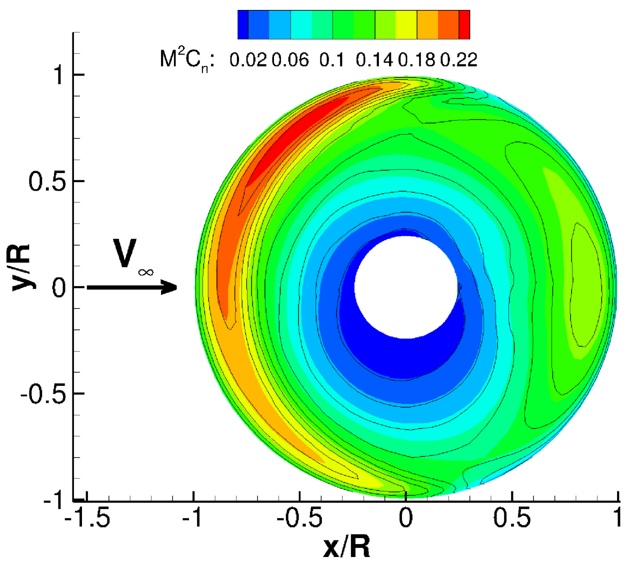
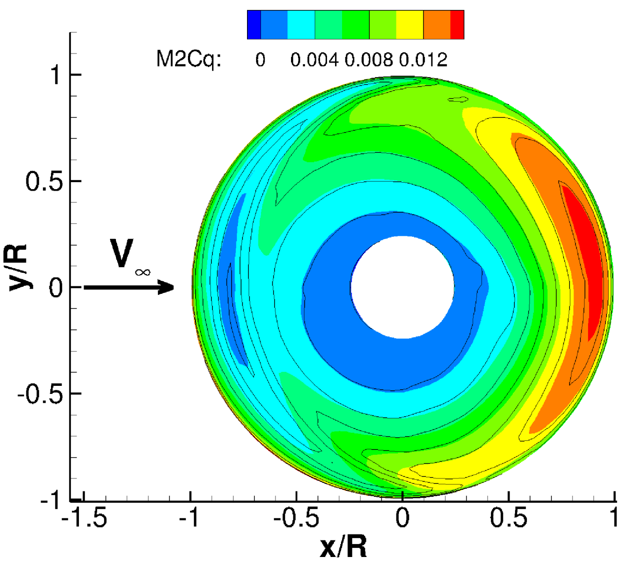


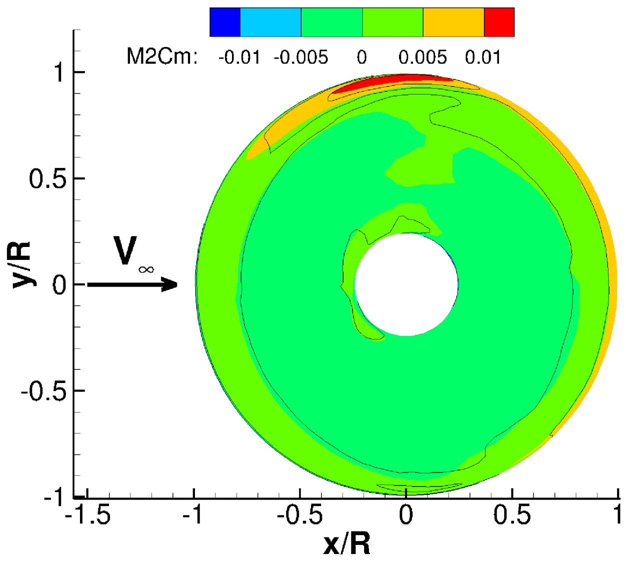
Figure 34: ROBIN, rotor fuselage case. Close-up of the quadruple overlap ROBIN topology at different time steps. (a) fuselage grid, (b) background grid, (c) fuselage geometry with disk grid and blades inside, (d) full overset grid topology, after hole cutting in three different time-steps. (e) and (f) show cell flags in the background grid.



(a)



(b)



c

Figure 35: ROBIN, rotor fuselage case. (a) Distribution of lift coefficient on the rotor disc, (b) torque coefficient, (c) blade pitching moment coefficient. The solid lines correspond to chimera and the contours to sliding plane solution.

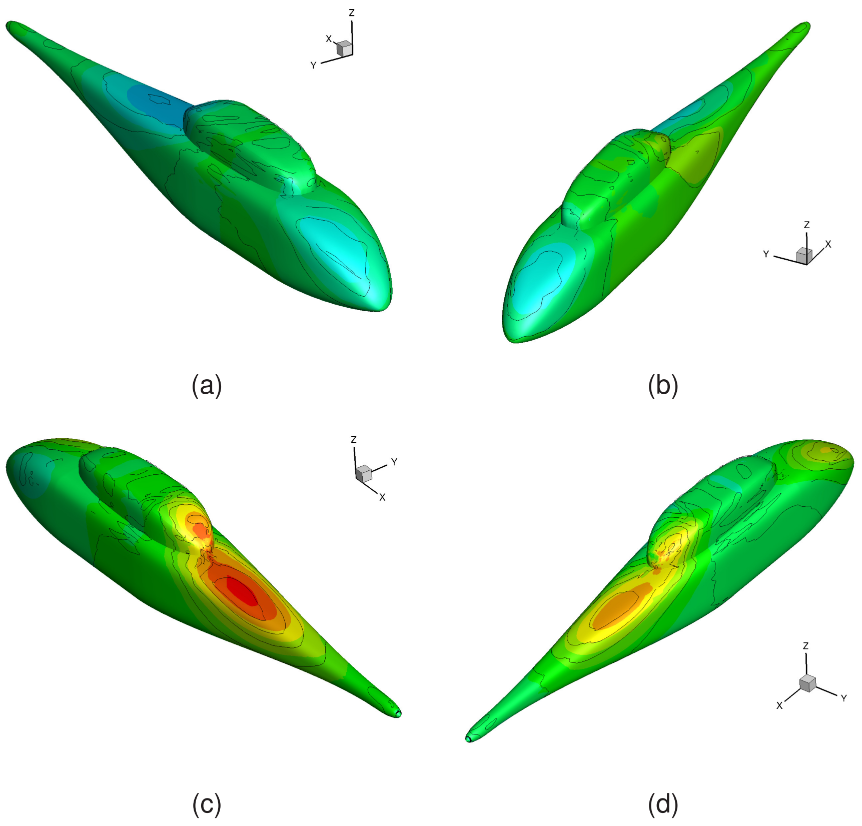


Figure 36: ROBIN, rotor fuselage case. (a) Unsteady pressure at the fourth rotor revolution, at the rotor position corresponding to $\Psi = 220$ degrees,(b) $\Psi = 240$ degrees,(c) $\Psi = 260$ degrees,(d) $\Psi = 360$degrees. The solid lines correspond to chimera and the contours to sliding plane solution.

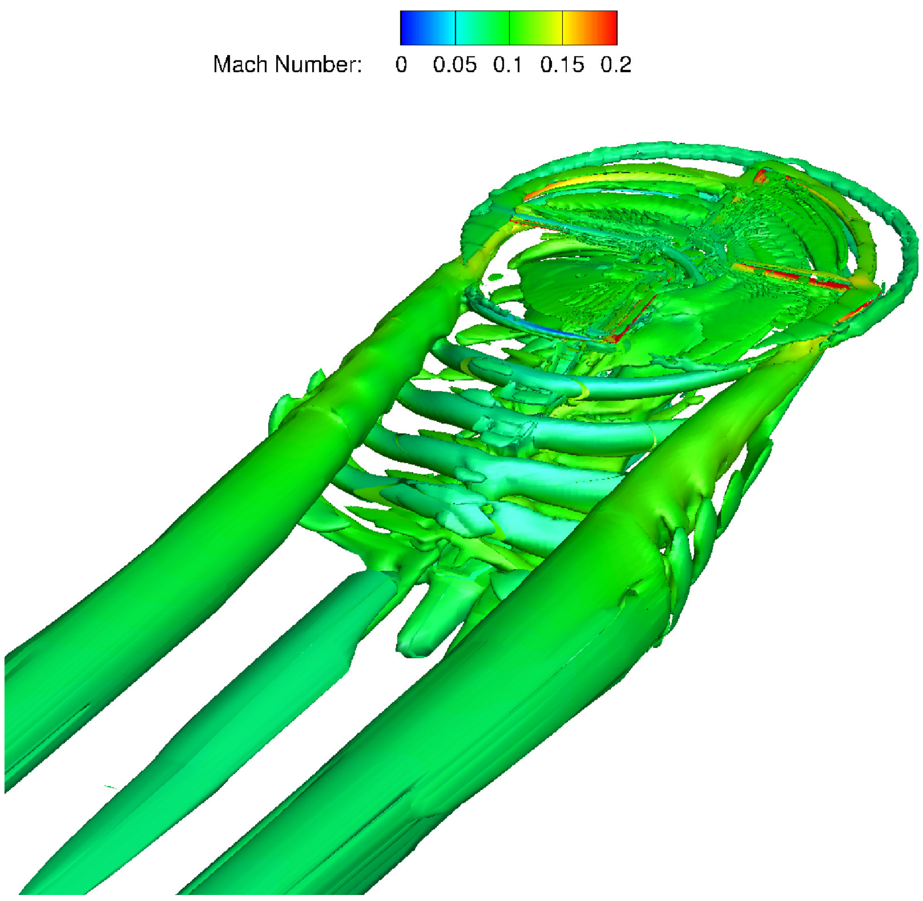


Figure 37: Wake visualisation: (a) Q criterion iso surfaces at $Q = 0.1$ with Mach number contours for ROBIN, (b) Q criterion iso surfaces at $Q = 0.1$ with Mach number contours for UH60 forward-flying rotor. In both cases, the solution was obtained using overset mesh method.