ID 99

TOWARDS CONSISTENT HYBRID OVERSET MESH METHODS FOR ROTORCRAFT CFD

M.Jarkowski[1], M.A.Woodgate[2], J. Rokicki[1], and G.N.Barakos[2]

[1] Institute of Aeronautics and Applied Mechanics
Warsaw University of Technology, 00-665, Poland
http://c-cfd.meil.pw.edu.pl/ccfd/
[2]CFD Laboratory, School of Engineering
University of Liverpool, L69 3GH, U.K.
http://www.liv.ac.uk/engdept
Email: G.Barakos@liverpool.ac.uk

Abstract

The use of overset grids is popular within the rotorcraft research community since such grids allow for the relative motion between the helicopter blades and fuselage to be accurately accounted for in Computational Fluid Dynamics methods. In this paper, a method is presented for treating overset grids within CFD codes. The method is simple to implement and is compatible with solvers enabled to compute on multi-block structured grids. The method is based on a hierarchy of searches that characterise mesh cells located in the overlap region between two structured blocks. The efficiency of the method relies in the utilisation of a search tree approach. Due to the efficiency of the algorithm the search for overlapping cells can be carried out on-the-fly while the CFD solver is computing the flow domain and is suitable for parallel execution. The method has been demonstrated for several flows ranging from simple aerofoils to rotor-body interaction cases. The paper presents and demonstrates the method and shows that it has low CPU overhead. It also highlights the limitations of the method and suggests remedies for improvement.

1 INTRODUCTION

The relative motion between helicopter rotor blades and fuselage leads naturally to the development of overset mesh methods for the analysis of helicopter flows. Consequently, the overset method is relatively well established in the CFD research and practice and is in use today by researchers at Universities and practising engineers in the industry [2, 3, 5, 8, 10, 15]. Despite its use and acceptance two main challenges related to the efficient implementation and accuracy of Chimera methods still remain: (a) The identification of overlapping structure (localisation of the interface points of one CFD mesh with respect to other meshes) for complex flow cases and (b) the implementation of consistent and conservative flux computation methods in the framework of the control volume formulation employed in CFD solvers. In this paper, progress made towards both challenges is described using the Helicopter Multi-Block method (HMB) of Liverpool to demonstrate the overset methods under development. The novel elements of the proposed method are a fast hierarchical search method for the localisation of the solution and the implementation of a consistent flux-reconstruction scheme within the framework of strongly implicit CFD methods. The method begins with identification of block-to-block overlap at each instance of an unsteady flow computation. A search for cell-to-cell overlap is then conducted to categorise CFD cells as normal, fringes or holes based on the nomenclature introduced in the literature. [12, 16] The key idea is to use the implicit indexing of the structured grids to identify each cell on the mesh and mark it as hole or fringe as well as identify a cloud of near-cell neighbours on overlapping domains. The CFD grids associated with major parts of the helicopter follow a hierarchical approach (eg. background grid, blade 1, blade 2, fuselage etc) which is compatible with the overall mesh system of HMB [13].

2 NUMERICAL METHODS

For the purposes of this work, the Helicopter Multi-Block CFD solver was modified to allow for efficient solution localisation and efficient computation of the flow solution in the mesh overlap regions. These two modifications were possible due to the data-structure of HMB that has enough flexibility to accommodate extensions to the solver without major re-writing of the core solver functions [18].

2.1 Solution Localisation

The method employed for solution localisation consists of three steps. The first step is to generate a minimum volume bounding box (MVBB) around each block in the mesh. This is a relative straightforward task and Figure 1 shows a typical block and the associated MVBB. This is followed by the second step that identifies all block-to-block overlaps for a given test case. Since memory in HMB is allocated per block, finding all the blocks with potential overlap provides a mechanism for allocating the minimum memory required and also restricting the necessary searches for the identification of the solution at cell-level. The third step is the identification of each cell on blocks with potential overlap. At the end of this step, each cell in the computational domain has been 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 HMB solver to read or can be calculated on-the-fly as the HMB solver is executed, for unsteady flow cases with moving girds where the flow solution must be localised and re-localised during the computations. It should be noted that

HMB merges this information into three classes, normal cells, (*"fringe","normal-cell"*), holes, (*"cell-in-solid"*, *"hole-cell-in-fluid"*, *"hole-cell-with-interpolated solution"*) and finally interpolation (*"hole-cell-with-interpolated solution"*) which are the only cells which require extra information associated to them.

2.1.1 Minimum Volume Bounding Box

The MBVV generation consists of the following steps:

1. For each block perform the following operations:
   (a) Read the coordinates of the mesh points.
   (b) Calculate the centre of mass of each block.
   (c) Calculate the matrices for the second area moments about the origin and centre of mass.
   (d) Find the eigenvectors of the above matrix and create a rotation matrix (constructed out of the eigenvectors in rows).
   (e) Apply rotation to the block (about its centre of mass) and determine the co-ordinate ranges in x, y and z directions.
   (f) Free the memory allocated for the mesh point coordinates coordinates.
2. Write a box file for each block which contains the coordinates for the principal axes of inertia of each block as well as the x, y, z ranges of the MVBB rotated to align with the axes of the co-ordinate system.

   Some blocks halo cells may also require interpolation information from other levels so the MBVV is increased in size to include this possible case in the Block-to-Block overlap calculation.

2.1.2 Identification of Block-to-Block Overlap

This step utilises the MBVV information to identify which block overlaps with each other block in the mesh. The method consists of the following steps:

1. Read the number of blocks in each grid component (fuselage, blade, flap etc) and the total number of blocks ($nblt$)
2. Allocate memory for the external 6 faces of each block and read all mesh points on solid walls.
3. Find the cell with the smaller size in each block. This value will be utilised for for local refinement close to the block edges.
4. Read and store the nodes on the edges of each block.
5. Convert the node information to cell-centre information and store it in the HMB data structure. Only external block faces are stored.
6. Use the MVBB information and check if any boundary nodes of a block are within the MVBB range of any of the other blocks in the mesh.
7. To account for inclusions if block i overlaps with block j then block j must overlap also with block i.
8. Store the block-to-block overlap information.

This algorithm is very efficient and data parallel, however, it is always optimistic in the generated block-to-block overlap information. Consequently, some of the the block overlaps that are to be tested for cell overlap will produce no results.

2.1.3 Cell Overlap Search

This part of the method also deals with hole cutting and employs an algorithm that will perform $n^{log(n)}$ searches between two overlapping blocks. The interpolation scheme can be zero order though a second order method is used for compatibility with the spatial scheme of HMB. Hole cutting is based on mesh hierarchy. At first, hole cells within solid boundaries are found based on the information stored in HMB for solid boundary cells. Node-in-solid is identified by the vector product criterion, which has proved to be accurate for all test cases. However some grids with very high aspect ratios can possibly generate inaccurate results. In future an additional method (solid bounding box) will be developed to assure correct flags for the cells in solid. The hierarchy of grids is usually defined with a special input file. Before any node belonging to any block of a mesh at a level in the hierarchy is localised with respect to any other level, an in-house range-tree is used, to determine in which cells of a block the node is potentially located. Range tree works like a kd-tree, but it returns a range(s) (cell's axis-aligned bounding box), in which the point is located instead of the nearest neighbour of the point. The in-house range-tree algorithm is also capable of finding the nearest neighbour, but a simple kd-tree strategy is not optimally efficient. For this reason, at the nearest point search stage, a binary Approximate-Nearest-Neighbour library is used, which integrates a kd-tree with other nearest-neighbour strategies. The library has been extensively tested and well documented [9]. An exact arithmetics library is used to determine if a point is inside a cell. This library is documented in reference [4].

Load balancing is essential for the efficiency of the search algorithm and in this work, intersecting blocks are grouped into sets with one block considered to be a host, and the rest, referred to as neighbours. Each block can be a host only once, but can be a neighbour more than once as shown in Figure 2. This process forms a set of searches that are then distributed between the processors.

The grids used in HMB are structured and body-fitted with hexahedral elements, numbered as presented in Figure 4. Each cell wall can be treated as a piecewise linear shape made of four triangles with a common apex located in the wall's centroid (Figure 5). Each of the 24 triangles constructed as described above is a base of a tetrahedron. A hexahedral cell is replaced by a set of 24 tetrahedra with a common apex in the cell centroid. This operation is consistent in terms of mesh volume as the position of wall centroid is independent of the side it is calculated from (inside or outside). Moreover, unlike a hexahedron, a tetrahedron has a simple linear transformation into a normalised shape (Equation 1):

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

where:
$i = 0, 1, 2, 3$.

At the stage of node localisation within the cells of another level, the position of the node at hand, is checked and flagged as inside of outside with respect to any other another grid level. For an exact and consistent check this procedure is based on tetrahedra. The position of a node with respect to the walls of the tetrahedra is checked using Schewchuk's exact arithmetics library [4]. In general, the position of point P with respect to a plane (below or above) is defined by A, B and B can be checked as follows (see equation 2):

$$\begin{vmatrix} x_A - x_P & x_B - x_P & x_C - x_P \\ y_A - y_P & y_B - y_P & y_C - y_P \\ z_A - z_P & z_B - z_P & z_C - z_P \end{vmatrix} \qquad (2)$$

The details on this method and assurance for the correct sign of the determinant can be found in [14].

After the node localisation the cells are flagged according to the following rules:

1. A cell is marked as a *hole* if all 8 nodes are inside another grid level.
2. A cell is marked as a *fringe* if at least one and less than 8 nodes of a cell are inside another grid level.

Fringe cells, shown in red in Figure 3 are bounded by both normal cells, shown in green, and hole cells, shown in cyan. However to be to calculate the cell residual of any given fringe cell flow solution must be available for the whole of the residuals stencil. In HMB case, this means 2 layers of cells from each face. To achieve this those cell as flagged as *interpolation* cells, shown in yellow, for which an interpolation stencil is to be found.

Depending on the position of the cell in the grid hierarchy, an interpolation stencil may consist of:

1. one nearest cell (zero order interpolation), or
2. 10 to 15 nearest cells - cloud of the nearest centroids(2^{nd} order interpolation based on Least Squares approach), or
3. 7 cells, for the case of an *interpolation* cell with its adjacent cells (2^{nd} order without cross terms interpolation based on the reconstruction of flow variable distribution within a cell).

An outline of the cell localisation process is given below. The list includes only the major steps that have to be performed.

1. Read number of levels (nlv) and total number of blocks ($nblt$) from the input files of the HMB solver.
2. Create a block-limiter vector, which is a map between a global (0 to $nblt$) and local (associated with levels) block numbering
3. Allocate memory for blocks. All blocks are stored in one-dimensional arrays and each processor stores one or more host blocks as well as all the blocks that belong to its group.
4. From the $.lev$ input file read the names of grid files of all the components of the mesh.
5. Read and store basic block information (but no mesh co-ordinates) for all blocks of the mesh.
6. Read the number of overlaps for each block from the $overlap.head$ file.
7. Read the mesh hierarchy file.
8. Read the data on the intersection - which blocks should be localised inside other blocks on the various levels of the hierarchy.
9. Calculate the load balancing of the cell localisation search.
10. Read the coordinates of all solid surfaces of the mesh that are to be localised by the current processor.
11. Localise nodes with respect to solid boundaries. The binary tree is used to find the nearest point of the solid for each node. Then the normal vector product criterion is used to determine if a node is inside or outside the solid body.
12. Flag the cells within a solid as *holes* (see Figure 6a).
13. Flag two rows of cells as holes and solids. This assures that the flow variables are never interpolated from a cell in direct proximity of a solid body (Figure 6b).
14. For any two blocks (A and B), which belong to different levels, and for which the overlap has been identified in the previous step do the following:
    (a) Localise against the host block all nodes from the group of neighbouring blocks.
    (b) Insert the nodes of all the neighbours in a group to a range tree.
    (c) Conduct a search loop using the range tree.
    (d) Flag each nodes as in or out of any cell in each neighbouring block.
15. For each cell in each block do the following:
    (a) Check how many nodes of the cell are inside any other cell. If none of them - flag the cell as *normal*, if all of them, flag the cell with *hole*. If more than one, but not all of the nodes of the cell are inside of another cell than the cell is flagged with *fringe*.
    (b) Flag all cells belonging to a Chimera boundary with a `CHIMERA_BOUNDARY` tag.
16. Find the donor cells for each cell that needs to have interpolated values.

17. For each interpolated cell find the nearest neighbour or the list of nearest neighbours according to the desired solution reconstruction method.

18. Write out TECPLOT files with the overlap information for inspection and visualisation.

19. Write out binary files that the HMB solver can use to restart or compute a solution on the overlapping mesh.

A sample of the outcome of the above process for a simple case of a regular background mesh and a foreground mesh around a NACA0012 aerofoil can be seen in Figure 7(a). The foreground mesh is a higher in the hierarchy, as it is much finer. The region of the background grid which has an overlap with the foreground mesh is treated as a hole (Figure 7b).

2.2 Helicopter Multi-Block solver

The Helicopter Multi-Block(HMB) code, 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:

$$\frac{d}{dt}\int_{V(t)} \vec{\mathbf{w}} dV + \int_{\partial V(t)} \left(\vec{F}_i\left(\vec{\mathbf{w}}\right) - \vec{F}_v\left(\vec{\mathbf{w}}\right)\right)\vec{n} dS = \vec{S} \quad (3)$$

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:

$$\frac{\partial}{\partial t}\left(\mathbf{w}_{i,j,k} V_{i,j,k}\right) = -\mathbf{R}_{i,j,k}\left(\mathbf{w}_{i,j,k}\right) \quad (4)$$

where $\mathbf{w}$ represents the cell variables and $\mathbf{R}$ the residuals. i, j and k are the cell indices and $V_{i,j,k}$ is the cell volume. Osher's [11] 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 [1].

Multi-block structured meshes are used for HMB. These meshes are generated using ICEM-Hexa™of Ansys. The multi-block topology allows for an easy sharing of the calculation load for parallel computing. Adding sliding meshes [17], as well as allowing for mesh overlap makes the HMB 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 HMB, hallo 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 hallo cells, its solution can be updated and then shared with other neighbouring blocks. In the strongly implicit HMB method, only the preconditioner employed for the solution of the linear system of equations is decoupled between blocks. 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.

3 RESULTS AND DISCUSSION

To evaluate and demonstrate the overlap mesh capability in HMB, several test cases have been considered. Figure 7 shows a simple case for the flow around a NACA0012 aerofoil. A mesh is constructed consisting of a background grid that covers the computational domain and a foreground mesh around the aerofoil. For this case, holes can be avoided by removing from the computation the background blocks that are completely covered by the foreground mesh. This is shown in Figure 7(b). The remaining blocks have a minimal overlap and a sample set of isobars of the solution is shown in Figures 7(c) and (d). The isobars are computed on both background and foreground grids to show the continuity in the solution on the overlap region. Due to the simplicity of this test case, it was only used for an initial check of the treatment of halo-cells in HMB and was then replaced by a rather more complex version shown in Figure 8. For this configuration, the background grid has no overlap with the foreground over a large central section of the domain and consequently certain cells were identified as holes. For the foreground mesh, Figure 8 (a) shows the location of the fringe cells that are to be computed using information from the background grid and vice-versa. The cells marked in blue colour are holes and their values will not be updated by the solution process. Figure 8 (b) shows the reverse situation with the solution localisation for the foreground mesh. The fringe cells were quickly identified and are shown in red colour.

Although the previous cases were useful for the initial stages of the method development, were restricted to 2D. The next test case considered a simple wing based on the NACA0015 aerofoil with an aspect ratio of 6.6. The configuration is shown in Figure 9. The case consists of a regular background grid and a standard C-type mesh around the wing with extruded tip blocks. The solution localisation is also shown and this time a more detailed identification method is used. The method not only detects holes, normal cells and fringes but also shows the interpolation cells in cyan colour. The cells in blue are identified as "cell-inside-solid" and are treated like holes, the cyan cells are "holes-inside-fluid" and are also treated like holes. The yellow cells are "holes-in-need-of-data" and their values must be populated from other grids in the solution since these are to be used for the reconstruction of fluxes. The red cells are fringes and the green

cells are normal cells. Several planes of the background mesh are extracted and shown highlighting the regular localisation pattern obtained along the mesh. The total size of cells for this case was more than 4 millions and the search took less than a minute on a Pentium 4 computer. The solution shows the propagation of the wake generated around the wing on the background mesh. Due to the regular mesh employed, the wake and tip vortices were very well preserved. This is a potential advantage of the overset grids that allow for the best numerical properties of the underlying scheme if regular grids are used.

Following from the wing case, a rotor in hover was considered. This case is similar to the wing case though it has a different background grid. The ONERA 7AD rotor was used and a quarter of the rotor was considered. For this case, the block-to-block overlap detected some extra blocks with potential overlapping cells. This was due to the relative high aspect ratio of the blade and the large blocks that were used on the background grid. Regardless, the cell localisation was performed in a very efficient fashion and is shown in Figure 10. The solution captured well the blade loading that is compared on Figure 10(d) with a result obtained using a matched grid.

To demonstrate that potential of the method to deal with more complex multi-block topologies that don't follow the pattern usually employed with HMB, the case of a missile inside a weapon bay was considered [7]. Figure 11 shows the employed topologies, the solution localisation and sample results. The localisation pattern on the background mesh shows the extend of the hole cells and the relatively complex extension of the cut near the fins of the store. The final solution shows no discontinuities near the boundary though it revealed limitations of visualising the overlap region where rendering of one solution over another must be carried out. For this case the output from HMB was written in a finite element format to allow for the irregular regions between the overlaps to be properly visualised. The time for the solution localisation is reported on Table 1 and the results show that 2 minutes were enough for this complex case.

The final case considered in this paper is rotor-body configuration of ROBIN [6] that has been used in the past for the validation of the HMB method. For this configuration, a typical O-grid is used around the fuselage. The blades use C-type grids with a fine mesh around the tips. The complete mesh contained more than 12 million cells and was localised in 240 seconds. The result of the solution localisation and a solution for the surface pressure on the fuselage can be seen in Figure 12.

This set of test cases is deemed adequate for the initial validation of the overset method in HMB and the results show that the employed scheme provides a good starting point for a production version. The accuracy of the solutions should be further compared against benchmark HMB solutions on matched multi-block grids to identify the limitations of the proposed flux calculation, implicit solver and solution localisation.

4 Conclusions and Future Work

In this paper a method was presented and demonstrated for using overset grids within CFD solvers. A hierarchical approach was used that is compatible with the HMB CFD solver and resulted in fast turn-around times for the localisation of the solution on relatively complex meshes. The method had enough flexibility to allow for arbitrary mesh overlaps and is designed with parallel execution in mind. The key ingredient to the method is the use of a tree-structure to represent the mesh and facilitate the localisation combined with an efficient local search for cell overlap. Once the solution is localised the CFD solver requires minimal modifications with respect to the computation on a fully matched grid. This is due to the employed facility for populating the halo cells of the mesh with the most appropriate values from the localised solution. The method was first used for simple cases like flows around aerofoils and progressively was demonstrated for 3D flows like flows around wings, rotors in hover and rotor-body configurations. The overlap region was the focus of attention to assess the accuracy of the method and the continuity of solution across the overlap region. The results showed minimal deterioration of the solution near the overlap region and the rate of convergence of the solver was not substantially influenced by the decoupled approach adopted for the implicit solution in the overlap region. This encouraging result shows the potential of the method. On the other hand, there are several areas where the method could be improved. This includes the use of more efficient memory structures, the demonstration of the scaling of the method for massively parallel computations, the treatment of overlapping solids where no combination of cells on either grid approximate the intersection accurately and the treatment of cells that appear and dissappear from hole status. This is last topic is important since there are possible configurations where this situation occurs near active flaps, rotor hub and pitch link assemblies as well as retractable undercarriage.

References

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

[2] M.J. Berger. On Conservation at grid interfaces. *SIAM Journal*, 24(5):967–984, 1987.

[3] L. Cambier and J.-P. Veuillot. Status of the elsa cfd software for flow simulation and multidisciplinary applications. 46th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, USA, January 7-10 2008. AIAA-2008-664.

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

[5] X. Juvigny G. Jeanfaivre and C. Benoit. Parallle Chimera Computations of Helicopter Flows. ICAS2004 International Congress of the Aeronautical Sciences, 2004. .

[6] S. L. Althoff J.D. Berry. Computing Induced Velocity Pertubations Due to a Helicopter Fuselage in a Fresstream. Technical Report TM-4113, NASA, 1989.

[7] S. Lawson and G. Barakos. Review of numerical simulations for high-speed, turbulent cavity flows. *Progress in Aerospace Sciences*, 47(3):186–216, April 2011.

[8] A. Madrane, R. Heinrich, and T. Gerhold. Implementation of the chimera method in the unstructured hybrid DLR finite volume TAU-Code. 6th Overset Composite Grid and Solution Technology Symposium, Ft. Walton Beach, Florida, USA, October 8-10 2002.

[9] D.M. Mount. ANN Porgramming Manual, http://www.cs.umd.edu/ mount/ANN/. Technical report, 2010.

[10] R.H. Nichols and P.G. Buning. User's manual for OVERFLOW 2.1. Technical report, NASA Langley Research Center, Hampton, VA, 2008. http://people.nas.nasa.gov/p̃ulliam/Overflow/Overflow_Manuals.html.

[11] 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.

[12] N.A. Petersson. Hole-Cutting for Three-Dimensional Overlapping Grids. *SIAM Journal on Scientific Computing*, 21(2):646–665, 1999.

[13] R. Steijl, G. 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.

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

[15] J. Sitaraman, M. Floros, A. Wissink, and M. Potsdam. Parallel domain connectivity algorithm for unsteady flow computations using overlapping and adaptive grids. *Journal of Computational Physics*, 229(12):4703–4723, June 2010.

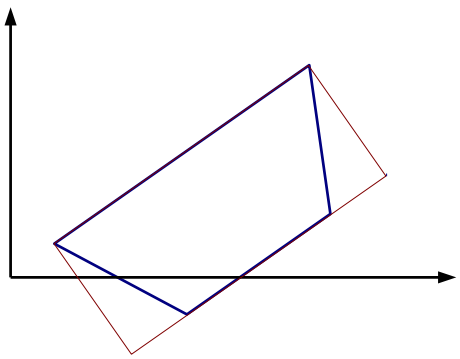
[16] B. Sjogreen and E. Part-Enander. Conservative and non-conservative interpolation between overlapping grids for finite volume solutions of hyperbolic problems. *Computers Fluids*, 23(3):551–574, 1993.

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

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

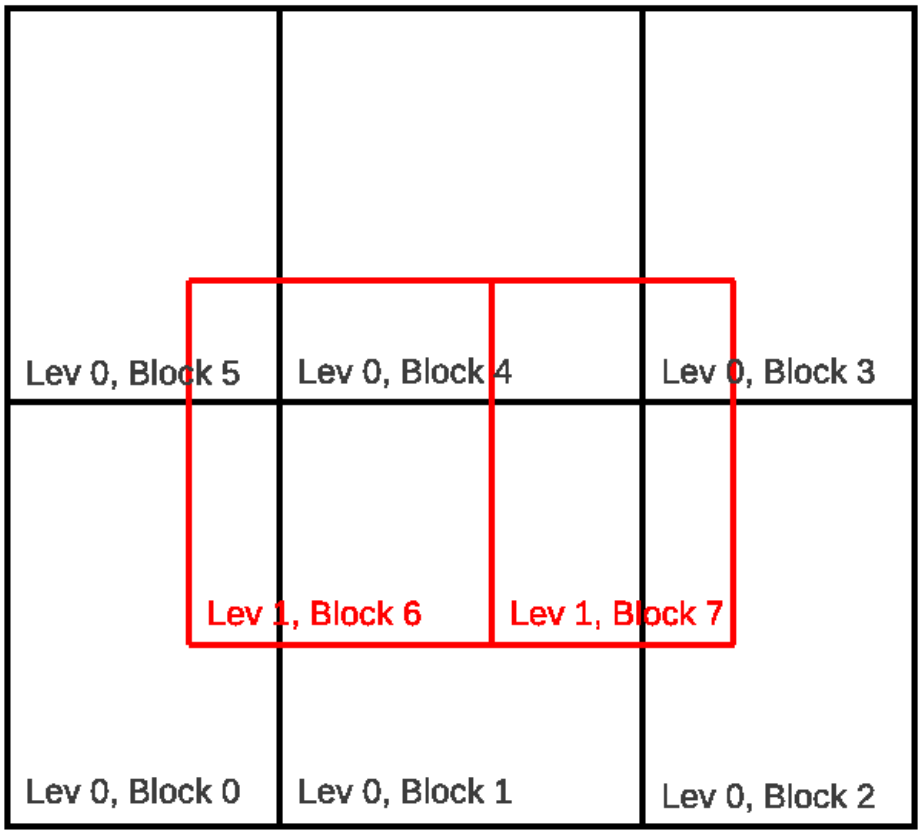
Test Case	Background mesh size (blocks)	Foreground mesh size (blocks)	MVBB	Block Overlap	Cell Search
Wing	2,488,320 (60)	1,511,424 (20)	25s	9s (6,20)	68s
Hover	2,592,000 (112)	350,892 (48)	21s	8s (98,48)	395s
Missile	5,000,448 (724)	3,015,680 (220)	53s	60s (42,220)	179s
Robin	7,190,604 (388)	1,458,528 (160)	55s	37s (91,160)	240s

Table 1: Sample CPU times on a single Pentium 4 processor for the solution localisation of the test cases.



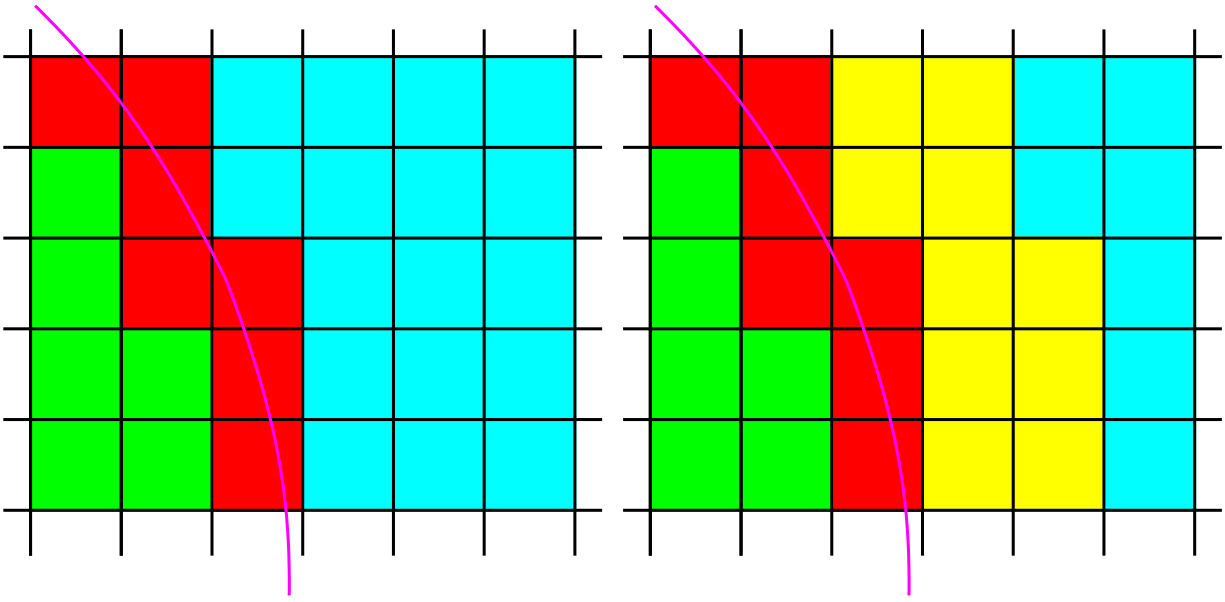
(a)

Figure 1: Minimum Volume Bounding Box (MBVV) around a 2D shape.



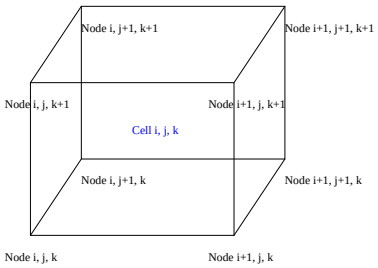
(a)

Figure 2: Multiblock topology with two different levels in 2D. In this case the following groups are created for load_balancing $Host : 0, Nbr : 6; Host : 1, Nbr : 6, 7; Host : 2, Nbr : 7; Host : 3, Nbr : 7;$ $Host : 4, Nbr : 6, 7; Host : 5, Nbr : 6; Host : 6, Nbr : 0, 1, 4, 5; Host : 7, Nbr : 1, 2, 3, 4$



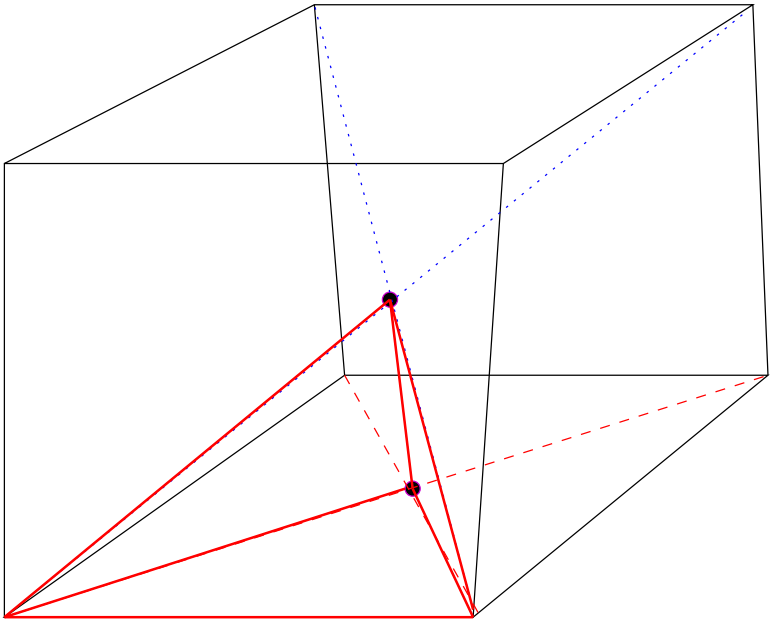
(a) (b)

Figure 3: (a) Shown the normal cell in green, the fringe cells in red, and the holes cells in cyan. (b) Shows the addition of interpolation cells in yellow to allow the cell residuals of all the fringe cells to be calculated.



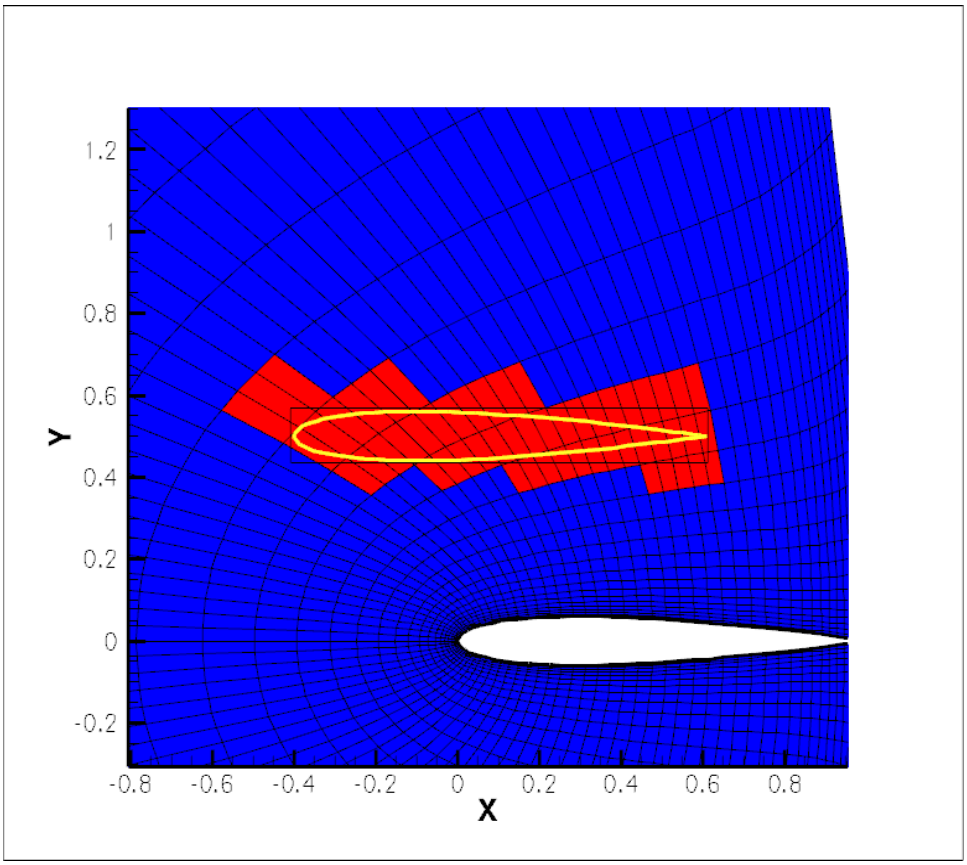
(a)

Figure 4: Cell and associated nodes nodes. All cells are unambiguously referred to by giving the lowest I, J and K indices of the nodes.

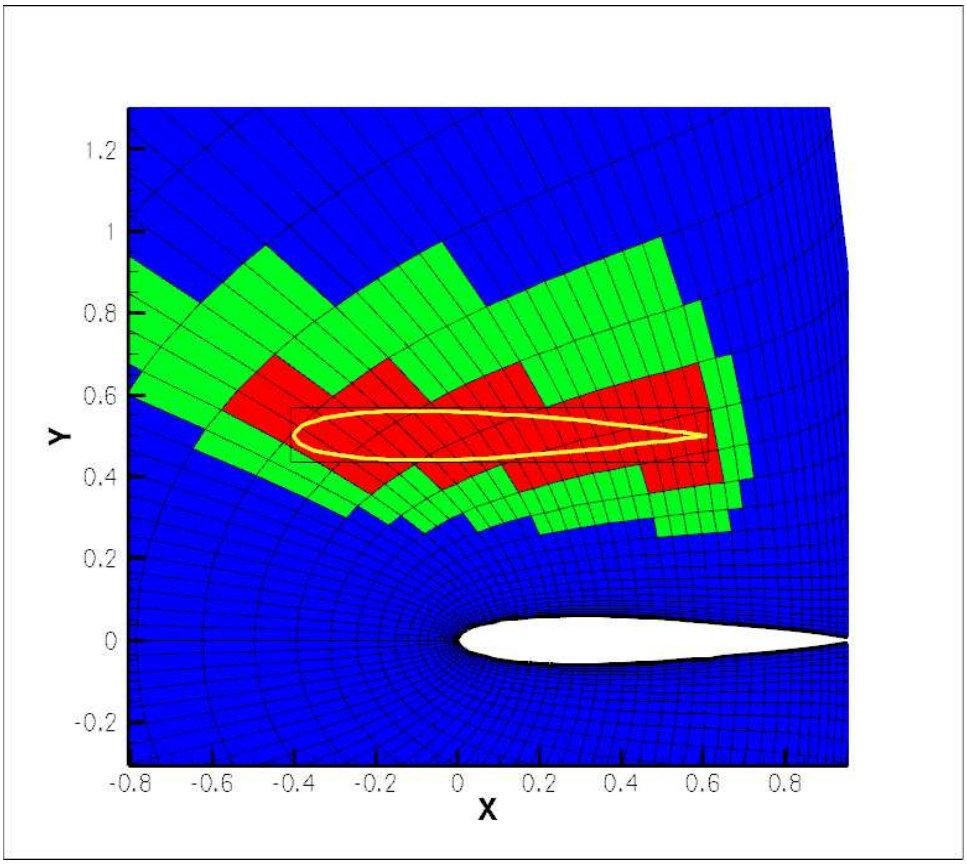


(a)

Figure 5: Hexahedron splitting into 24 tetrahedra. Each cell face is split into four triangles with a common apex (a-b-c). Each tetrahedron is thus split in 24 tetrahedra, with a common apex located at the cell centroid (a-b-c-d).



(a)



(b)

Figure 6: (a) Identification of solid_hole cells. In the second step (b) additional cells are flagged as solid holes. The number of cells flagged as solid holes in this step is dependent on the resolution of the grid which interpolates data from the current grid. The aim of this operation is to guarantee that the grid has at least two rows of interpolation cells outside of a solid body.

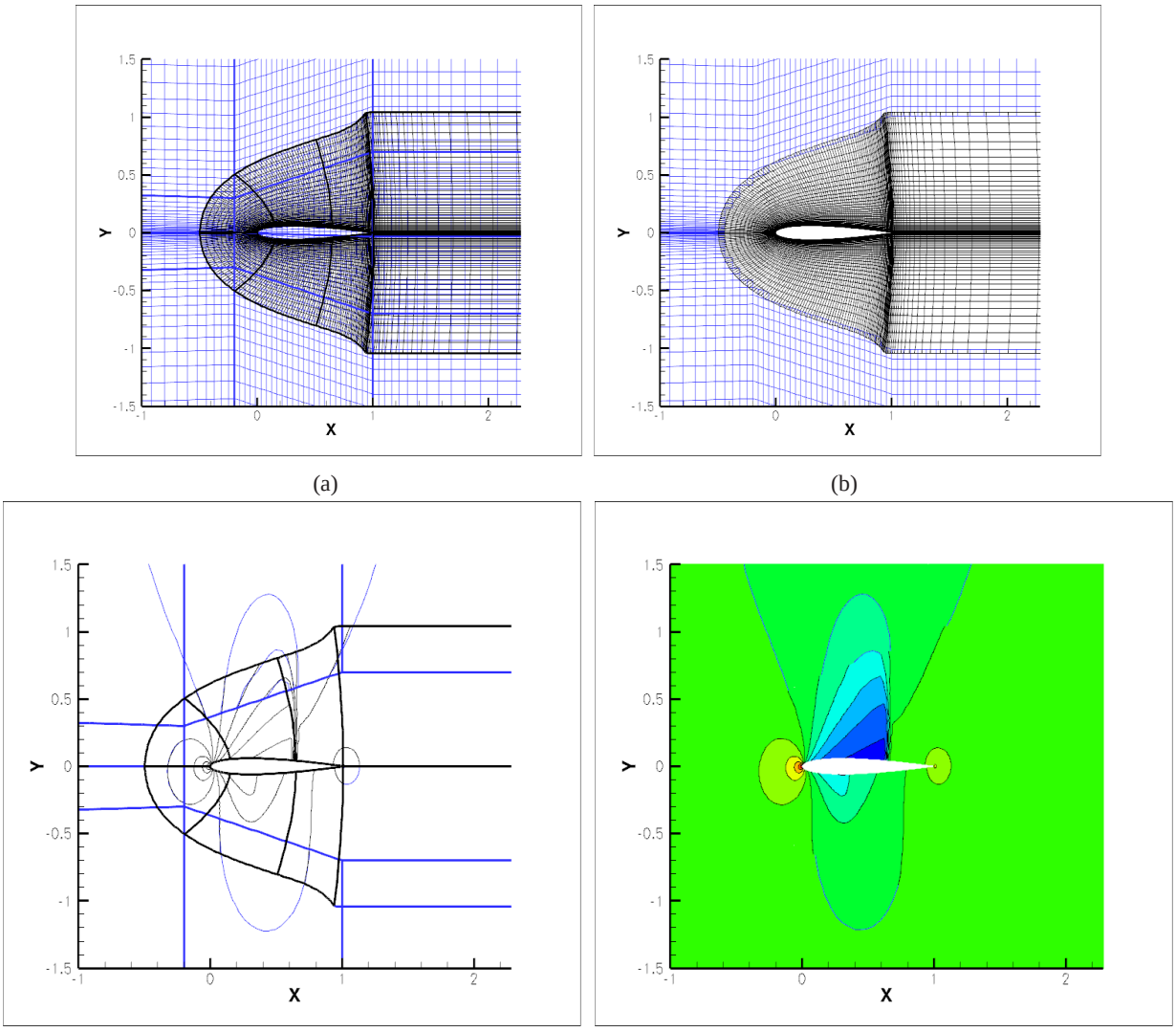
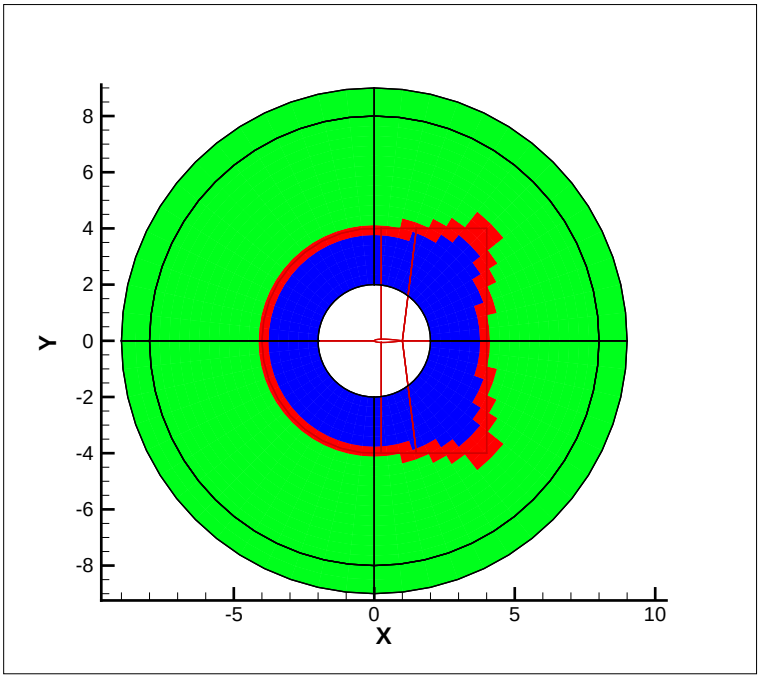
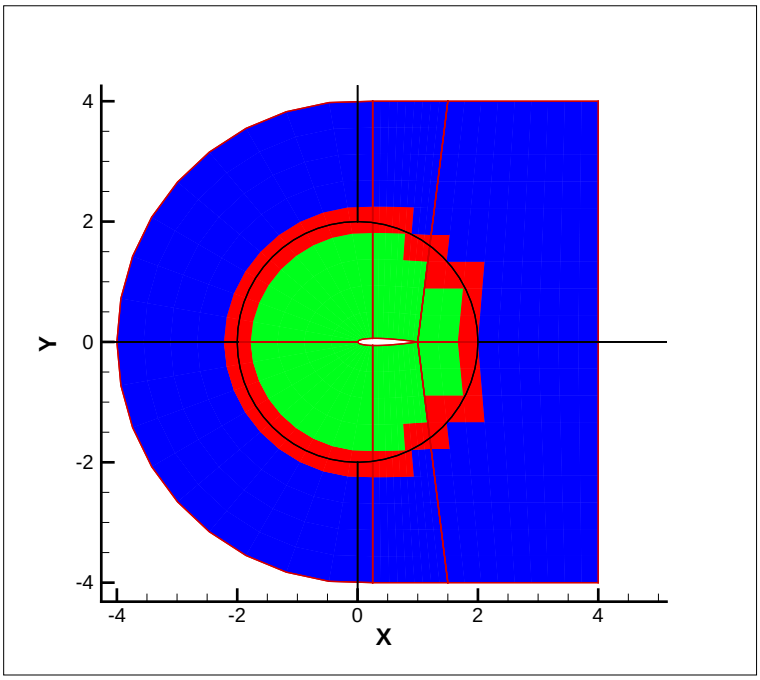


Figure 7: Simple 2D test case for a NACA0012 aerofoil. (a) The background mesh (Blue) is of a very simple topology and the foreground grid (Black) is of typical C-type (Black). (b) After removing the holes and localising the solution, isobars of the obtained solution are show (Mach 0.8, alpha 1.25 degrees).

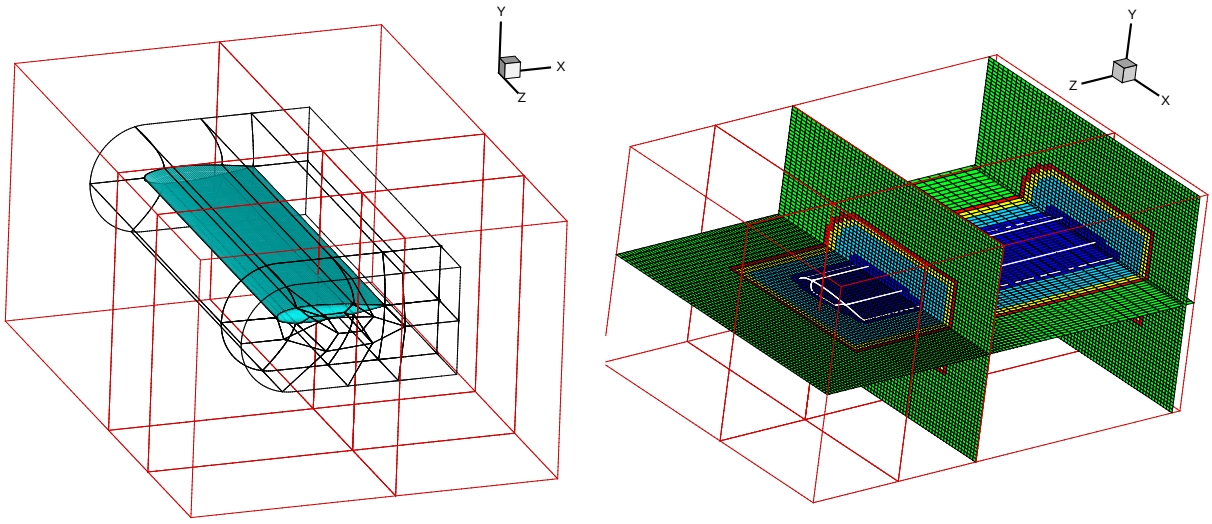


(a)



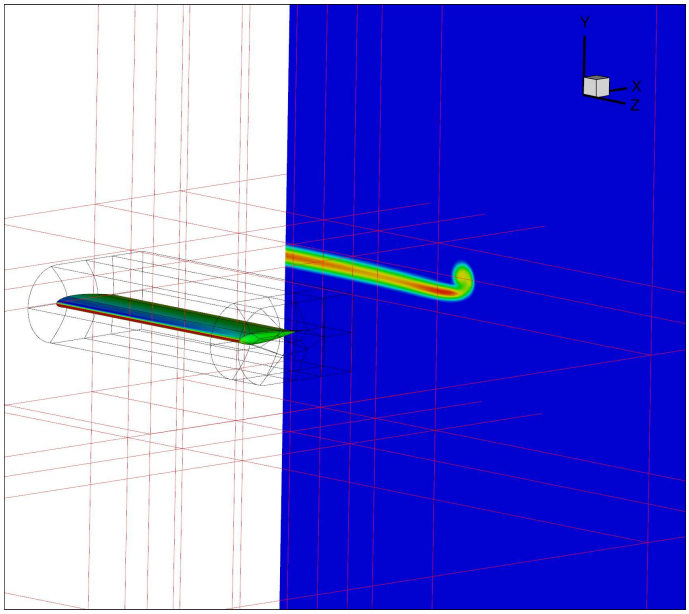
(b)

Figure 8: Simple 2D test case for a NACA0012 aerofoil. Normal cells (green), holes (blue) and fringes (red) located on (a) the background mesh and (b) the foreground mesh.



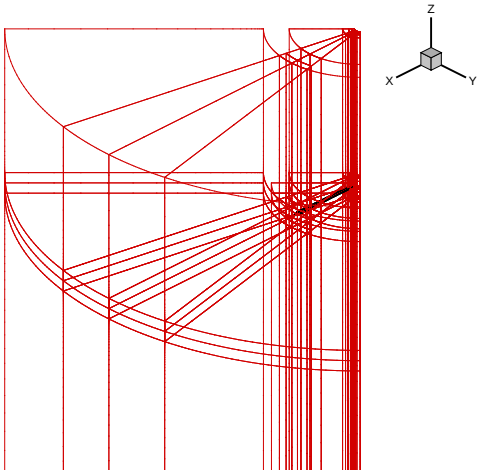
(a) Wing surface and body-fitted mesh topology

(b) Solution localisation

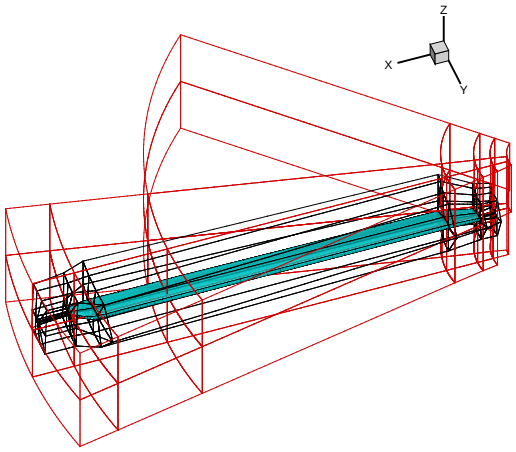


(c) Wake visualisation

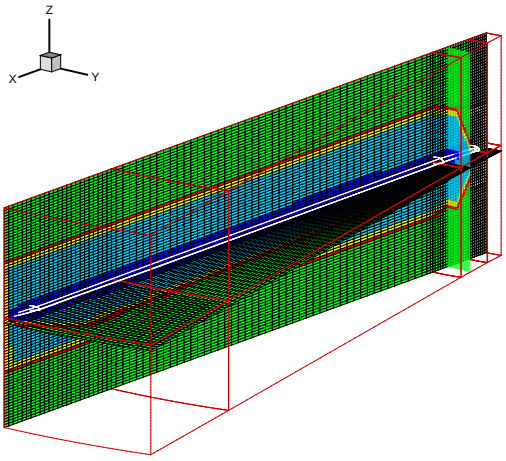
Figure 9: Demonstration of the method for a simple Wing case: Blue cells represent solid, cyan represent halo, yellow represent fringes and green are normal cells.



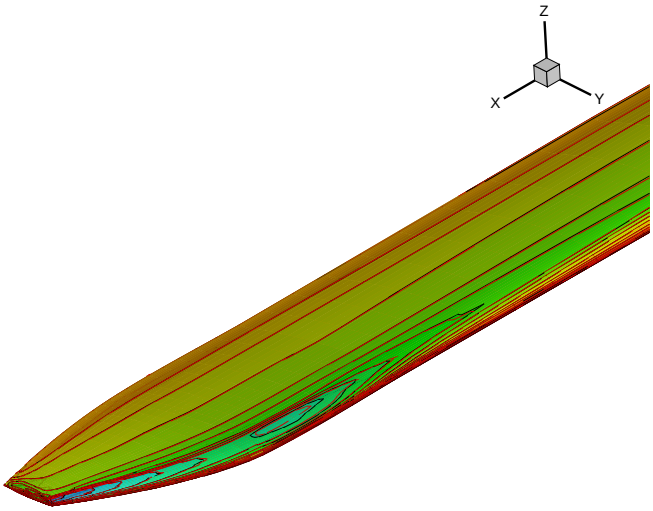
(a) Solution localisation - Block overlap search



(b) Block overlap near blade

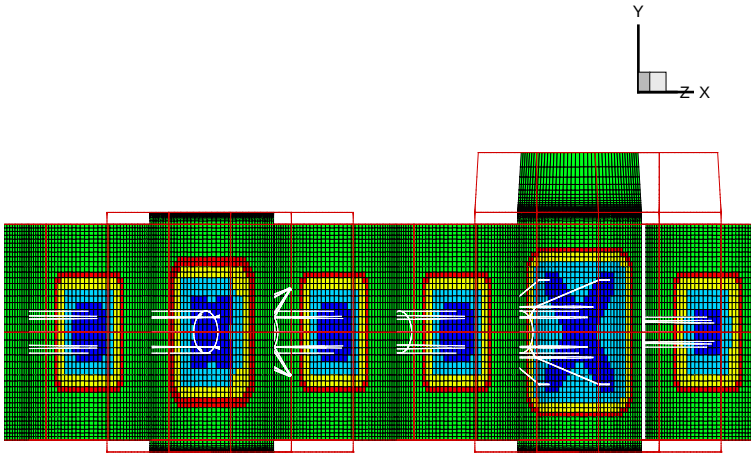


(c) Solution localisation near blade

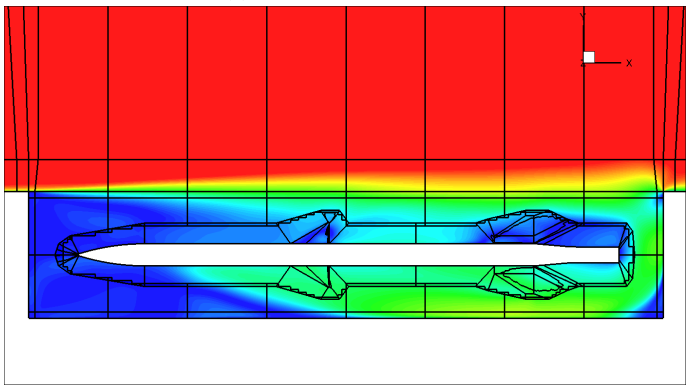


(d) Blade surface pressure

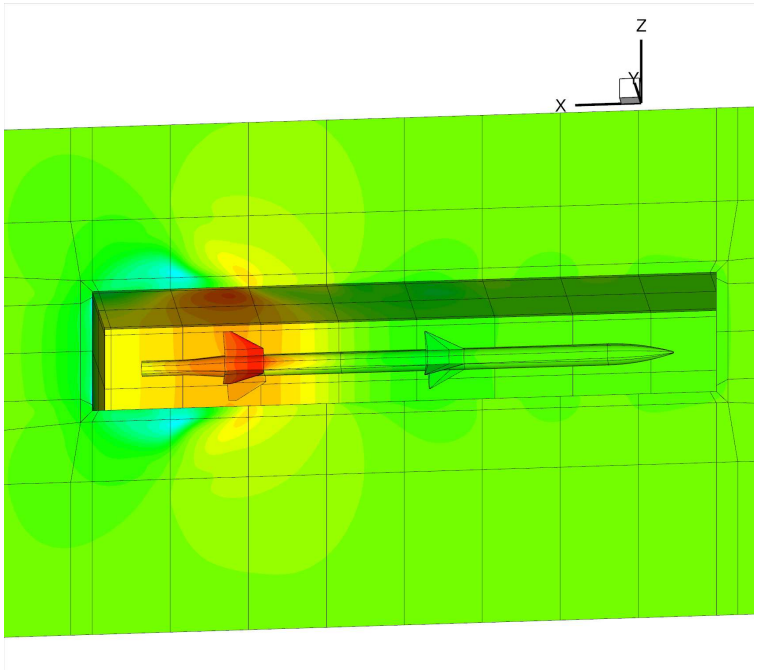
Figure 10: Solution localisation and sample results for the case of a hovering ONERA 7AD blade. The inviscid model was used for this computation and the near-blade grid was restricted to a single layer of blocks around the blade.



(a) Solution localisation

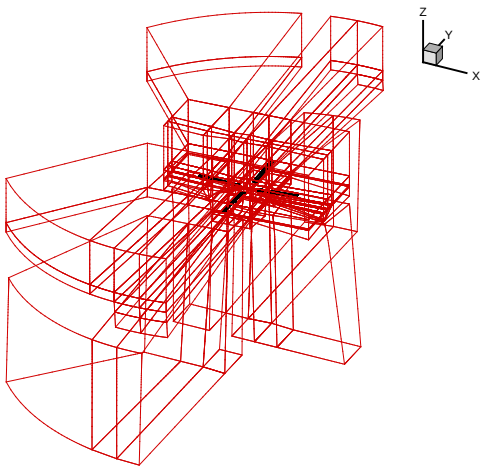


(b) Mach number field

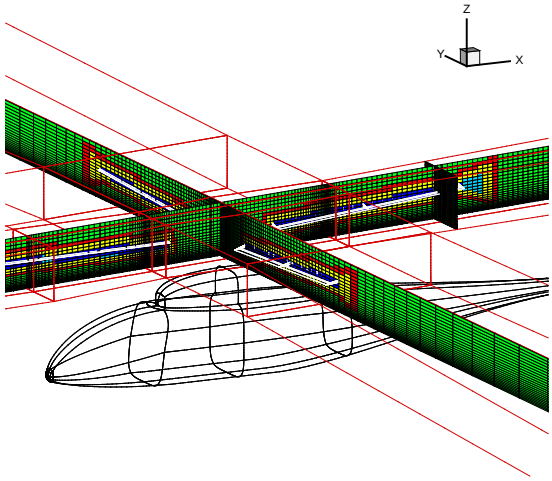


(c) Surface pressure

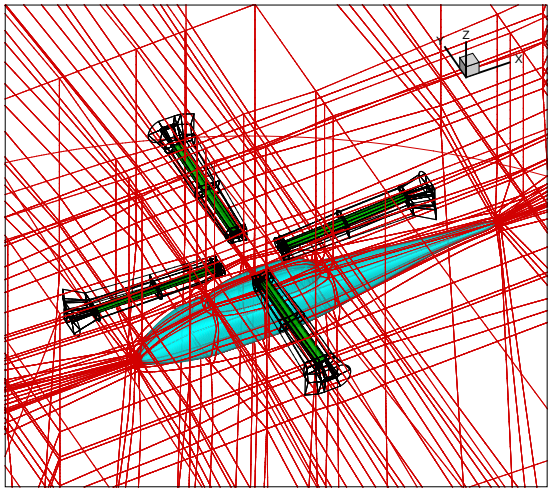
Figure 11: Solution localisation, and obtained Mach and pressure fields for the of a missile inside a weapon bay. The Spalart-Allmaras turbulence model was use for this calculation at a free-stream Mach number of 0.85 and for a cavity with length to depth ratio of 5.



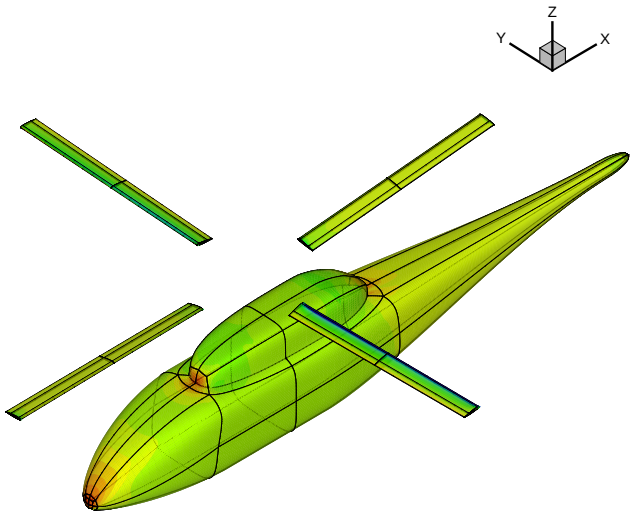
(a) Solution localisation - Block Overlap Search



(b) Solution localisation near rotor



(c) Mesh topologies



(d) Surface pressure on the fuselage and blades

Figure 12: (a) Block-to-block overlap, and (b) solution localisation for the ROBIN case. (c) Location of blade inside the background mesh, and (b) comparison between the overlap mesh results (red lines) and the matching mesh (colour contours) for the surface pressure on the 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.