DOI: 10.13009/EUCASS2023-037
Aerospace Europe Conference 2023 – 10TH EUCASS – 9TH CEAS

A local unstructured re-meshing technique for handling multi-body separation in 2D flows

C.A. Rasoni[⋆], *A. Assonitis*[⋆†], *R. Paciorri*[⋆]
[⋆] *University of Rome "La Sapienza"*
Via Eudossiana 18, 00184, Rome, Italy
rasoni.1528557@studenti.uniroma1.it · alessia.assonitis@uniroma1.it · renato.paciorri@uniroma1.it
A. Bonfiglioli[‡]
[‡] *School of Engineering, University of Basilicata*
V.le dell'Ateneo Lucano 10, Potenza, 85100, Italy
aldo.bonfiglioli@unibas.it
E. Cavallini[†], *S. Illiano*[†], *S. Ciabuschi*[†]
[†] *Italian Space Agency*
Via del Politecnico snc 00133 Rome, Italy
enrico.cavallini@asi.it · simone.illiano@asi.it · simone.ciabuschi@asi.it
[†]Corresponding author

Abstract

Numerical simulations of bodies experiencing large relative motion, as for instance the case of two separating bodies, is a challenging task for Computational Fluid Dynamic, because mesh deformation algorithms might be inapplicable and global re-meshing techniques could be computationally too expensive. For this reason, overset methods using overlapping grids are used to tackle these kind of problems. However, even though the overset techniques are very versatile, the transfer of information between overlapping meshes is an error source that can compromise the numerical solution quality. This work describes a new computational technique, that does not use overlapping meshes, but is instead based on local remeshing and makes only limited use of interpolation between different meshes. Algorithmic details of the aforementioned technique as well as two test-cases involving the separation of bodies in high-speed flows will be presented in this paper.

1. Introduction

The aerodynamic analysis of separating/colliding bodies, as well as rotating machinery, represents a tricky problem for the design of rockets, airplanes, propulsion and store separation systems: the presence of moving bodies which experience large displacements within the domain poses severe challenges to Computational Fluid Dynamic (CFD). In this case the grid deformation techniques[7] cannot be successfully applied, since large relative displacements among the bodies in a mesh that keeps the same topology cause large cell deformations that in general compromise the quality of the numerical solutions. In these circumstances, also global re-meshing techniques that regenerate the overall mesh are not suitable, since the time-dependent re-meshing of the domain can amount to a significant portion of the total computational cost and, moreover, global re-meshing also requires global interpolation of the numerical solution from the old mesh to the new one, thus increasing the computational cost and the numerical errors. For this reason, nowadays Overset methods[1] are very popular to handle problems that involve bodies in large relative motion: these techniques consist in independently generating the mesh covering the entire computational domain and the ones surrounding each of the moving bodies. Each grid can have different resolution, topology, and boundary conditions and, above all, can move independently of the others. By doing so, however, the flow variables need to be interpolated in regions where different grids overlap and at each time step. Although these methods are very simple and useful, their reliability is strictly linked to the data interpolation process. As a matter of fact, the interpolation process is a key factor in overset computations, because the choice of the interpolation scheme directly influences the accuracy of the solution.[8,9] Therefore, many studies can be found in the literature that asses the accuracy of overset methods[10] and investigate error propagation patterns and source.[5]
In this paper, an alternative to overset methods is proposed: it consists in a mesh generation technique for Moving Unstructured Meshes,[3] namely *MuMs*, that is based on local re-meshing and is capable to compute 2D flows featuring

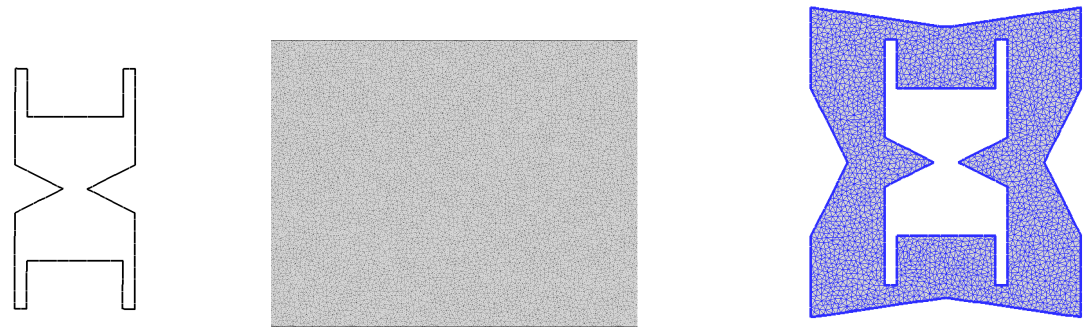
DOI: 10.13009/EUCASS2023-037

bodies in large relative motion. The key feature of *MuMs* is the absence of overlapping regions in the computational domain and the limited use of the solution interpolation.

2. Numerical method

The proposed numerical method consists in coupling an unstructured grid generator, which is used to create computational meshes around the moving bodies and within the fluid domain, with an unstructured, in-house gas-dynamic solver (`eulfs`) to compute the solution on the non-overlapping meshes created by the mesh generator. A comprehensive description of the `eulfs` code can be found in Refs.[2,3] Note, however, that the choice of the unstructured-grid CFD solver is not binding and, since it is invoked as a black box, it can be easily replaced by other solvers as long as the discretization is vertex-centred and the solver is based on an Arbitrary Lagrangian Eulerian formulation (ALE).
To illustrate the algorithmic features of the proposed mesh generation algorithm, let us consider a single moving object, such as the one displayed in Fig. 1a, moving inside the rectangular domain shown in Fig. 1b. The triangular mesh used discretize the rectangular domain is fixed in space, does not take into account the presence of the body and will be hereafter denoted as the *background* grid. A different grid, called the *mobile* grid, is generated around the body, as shown in Fig. 1c. This body-fitted mesh, which is therefore entirely independent of the background mesh, moves rigidly with the body. At time t, the body is located in an arbitrary position inside the domain, while over the time interval Δt, the body moves to a new location.

At time level t, we assume to know the solution vector, $\mathbf{Z}^t$, that is stored in the grid-points of both the background and mobile meshes. The process that leads from the known solution to the updated solution, $\mathbf{Z}^{t+\Delta t}$, at the subsequent time level can be split into the three steps that will be described in detail in the following sub-sections.



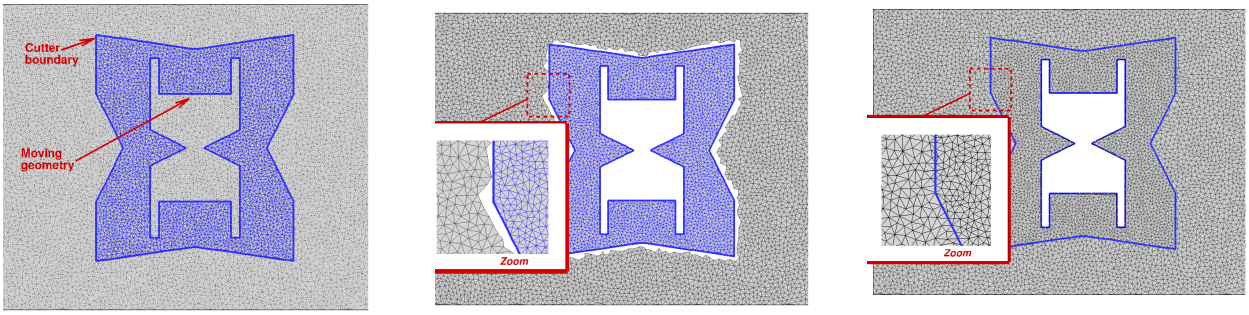
(a) (b) (c)

Figure 1: Moving geometry (a), background grid (b) and body-fitted mesh (c)

1. Generation of the *computational* mesh
   At time t the object occupies a specific position in the domain, therefore the mobile mesh covers a certain region of the background mesh as shown in Fig. 2a, which highlights both the outer boundary of the mobile mesh and the moving geometry. In the first step of the algorithm, all cells of the background mesh that have one or more of their vertices i) inside the outer boundary of the mobile mesh or ii) outside of it, but closer than a preset distance, are removed. The grid-points of the background triangulation that verify the former condition are identified using the Point-In-Polygon (PiP) algorithm,[6] which allows to determine whether a point falls within the polygon made up of the outer boundary of the mobile mesh. These vertices are then flagged for removal. Whenever the background grid contains a body, as will be illustrated in the test-cases in Sect.3, a similar check is performed to remove those nodes of the mobile mesh that fall inside the body. In this case the PiP algorithm considers the perimeter of the body in the background grid as a new polygon and therefore it is able to detect the mesh-points of the mobile mesh that are located inside this geometry.
   The reason for removing also the grid-points that meet the criterion ii) is that these nodes, if not removed, might be overcome by the mobile mesh as it moves from its location at time t to the one at time $t + \Delta t$.
   All the nodes of both the background mesh and the mobile mesh that have been removed are referred as "phantom" nodes. The cells of the background mesh with one or more phantom node are temporarily removed thus forming a hole, inside of which the mobile mesh is inserted, as shown in Fig. 2b. It can be seen that the background and mobile meshes are not overlapping and still disjoint at this stage. Local re-meshing is then applied only in the region bounded by the outer boundary of the mobile grid and by the inner boundary of the remaining

DOI: 10.13009/EUCASS2023-037

part of the background mesh. The two meshes are then mutually connected by means of a local constrained Delaunay triangulation (CDT), performed using the `triangle`[11] mesh generator, which does not introduce any additional grid-point (so-called Steiner points). Using the CDT, the edges that belong to the boundary of the hole of the background mesh and those that make up the outer boundary of the mobile mesh are constrained to be part of the final triangulation, which we refer as the *computational* mesh, see Fig. 2c. This new mesh is used to advance the solution from time t to time $t + \Delta t$. Before doing so, the solution vector $\mathbf{Z}^t$ that is stored in the grid-points of both the background and mobile meshes is easily transferred, without interpolating, to the vertices of the computational mesh because every node of the computational mesh belongs to either the background or mobile meshes.



(a) Background and mobile mesh superimposition

(b) Background cells removal and insertion of the mobile mesh

(c) Computational grid used in time evolution from t to $t + \Delta$t

Figure 2: Computational mesh generation in step 1.

2. Evolution of the solution on the computational mesh
   As stated before, the solution $\mathbf{Z}^t$ and the computational mesh at time t are supplied as input to the `eulfs` solver in order to advance the solution to the next time level. Since some of the nodes of the computational mesh, more precisely those that also belong to the mobile mesh, move with the body, a nodal grid velocity must also be supplied to the CFD code. More precisely, for each grid-point i of the computational mesh, the CFD code needs the nodal grid velocity w_i computed at time $t + \frac{\Delta t}{2}$, that is computed using the finite difference formula:

$$\frac{x_i^{t+\Delta t} - x_i^t}{\Delta t} = w_i^{t+\frac{\Delta t}{2}} \tag{1}$$

   The motion of the grid-points of the computational mesh causes a rigid motion of the triangles that also belong to the mobile mesh and a deformation of the cells that have been generated by the local CDT. The remaining cells of the computational mesh do not deform or move. The output of the CFD code is the updated solution $\mathbf{Z}^{t+1}$ within all grid-points of the computational mesh at time $t + \Delta t$.

3. Projection of the solution available on the computational mesh on the background and mobile meshes
   The computed solution at time $t + \Delta t$ has to be transferred from the computational mesh back to the background and mobile meshes. This is again trivial in those grid-points of the mobile and background meshes that belong to the computational mesh. It is only in the phantom nodes that the projection of the solution $\mathbf{Z}^{t+1}$ requires interpolation. The dependent variables of the phantom nodes need to be updated to time level $t + \Delta t$, because some of the phantom nodes might re-appear at a subsequent time level, due to the motion of the mobile mesh. The interpolation of the phantom nodes, which is made using the nodal values of the computational mesh, allows to compute the state of all phantom nodes that have been removed in step 1, except for those that are inside the moving geometry (or, eventually, inside the body contained in the background grid). At this stage, the numerical solution has correctly been updated at time level $t + \Delta t$ on both the mobile and background meshes, so that the next time level can be computed re-starting from the first step 1 of the algorithm.

3. Bodies separation in 2D high speed inviscid flows

In the following subsections two test-cases dealing with body separation in 2D high speed flows will be presented. These examples are used as benchmarks for the proposed numerical method to test the computational mesh-generation

DOI: 10.13009/EUCASS2023-037

process described in Sect.2. The first test-case deals with the separation of the first stage of a space-launcher during the first phase of the flight, whereas the second test-case deals with the unhooking of an external tank from a NACA0012 airfoil in transonic flight.

3.1 Space launcher stages separation

The first test case simulates the first stage separation from a space launcher that takes place during the early stage of the ascent. To minimize the computational cost, a simplified geometry, sketched in Fig. 3, was considered: it consists of a first stage (depicted using blue solid lines in Fig. 3) and of a portion of the second stage, marked by red lines. Indeed, when considering a null angle of attack and a space launcher simple geometry, the choice of using a reduced domain is reasonable as first approximation.
Then, two unstructured grids were generated. The former covers the overall computational domain and also contains part of the second stage; it is displayed in Fig. 4a and it is treated as the background grid by the *MuMs* algorithm. The latter mesh, shown in Fig. 4b, surrounds the first stage; this is the mobile mesh, because it is in relative motion w.r.t. the second stage of the space launcher. More precisely, we impose that the first stage translates along the launcher symmetry axis with constant velocity v =0.5 m/s, starting from the position shown in Fig. 4c.

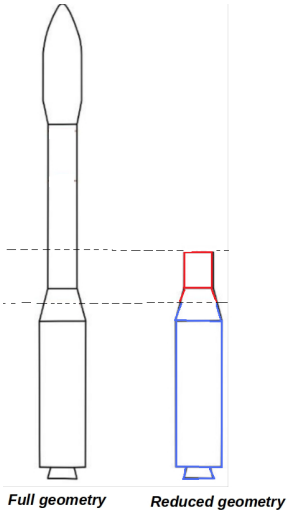


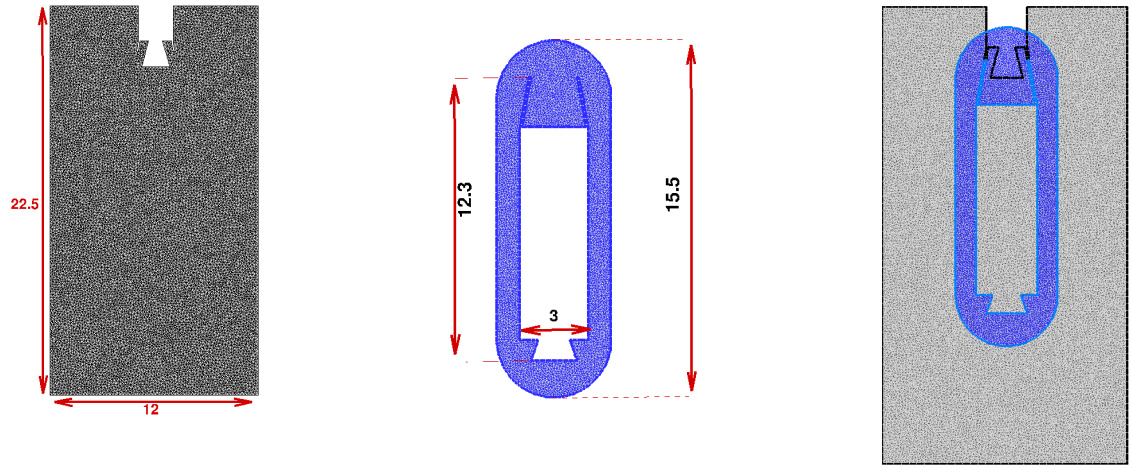
Figure 3: Space launcher sketch: full and reduced geometry

The flow past the space-launcher is supersonic (M = 2) and the angle of attack is zero. Moreover, stage separation occurs at an altitude of 50 km. As the first stage moves away from the space-launcher, the *MuMs* algorithm concatenates the background and mobile grids as shown in Fig. 5: in particular, the smaller frames of Figs. 5a and 5b highlight how the mobile grid is constrained to be a part of the computational grid and no overlapping occurs. Figure 6 shows the dimensionless density flow-field during stage separation at three different time instants computed by coupling *MuMs* with the `eulfs` solver.

3.2 External tank unhooking from a wing

In this subsection the MuMs algorithm is used to simulate the release of an external tank from a NACA0012 airfoil flying at transonic (M = 0.8) speed. More specifically, this test-case is inspired by a similar one reported by the userguide of the commercial code CFD++,[4] even if the computational domain and flow conditions are different.
Fig. 8a shows the background grid, which contains the airfoil: it consists in a square domain, whose side is equal to 10 airfoil chords. The mobile grid generated around the tank is displayed in Fig. 8b: also in this case, the tank length is made dimensionless using the airfoil chord. The initial tank position relative to the airfoil is displayed in frame 8c. Then, the tank moves away from the NACA0012 profile following the user-prescribed trajectory given by Eq. (2), where the ($x_0 = 0.4, y_0 = -0.1$) coordinates refer to the position of the tank leading edge at $t = 0$, shown in Fig. 7.

DOI: 10.13009/EUCASS2023-037

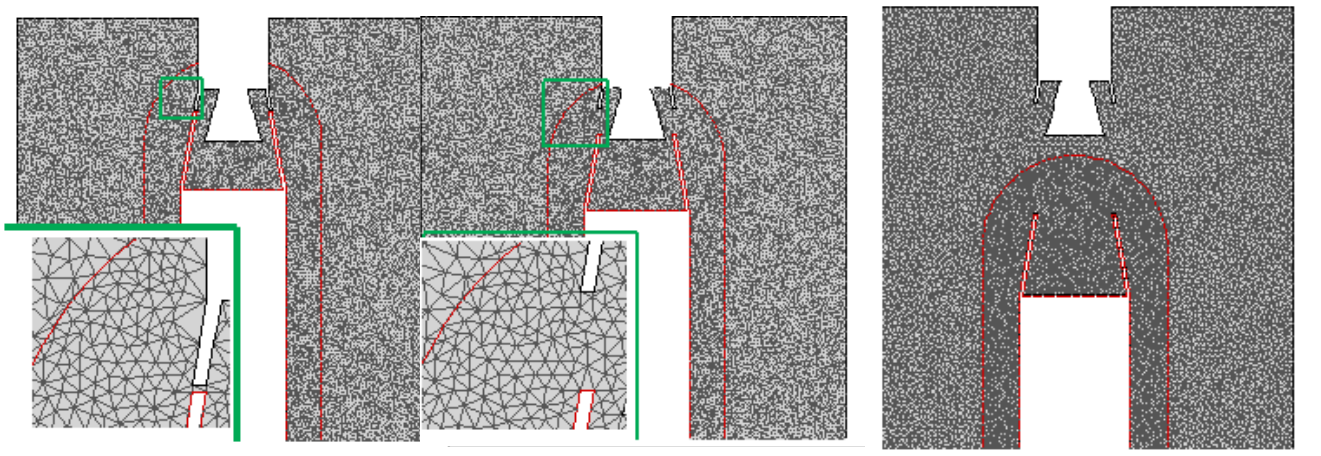


(a) Background grid (10688 nodes, 20932 cells)

(b) Mobile mesh (3689 nodes, 6825 cells)

(c) Store location at t=0

Figure 4: Space launcher stage separation: grids and first stage initial position.



(a) (b) (c)

Figure 5: Computational grid detail during first stage motion

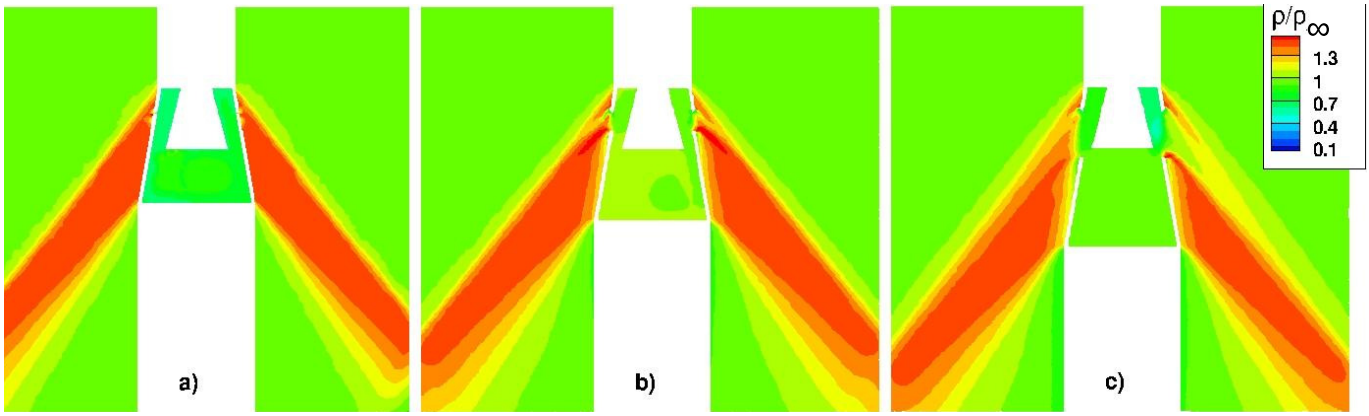


Figure 6: Density flow-field during first stage motion

DOI: 10.13009/EUCASS2023-037

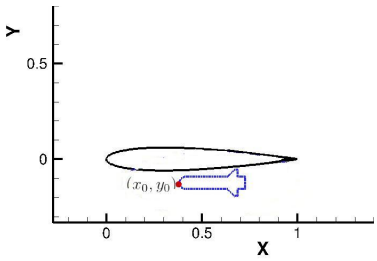
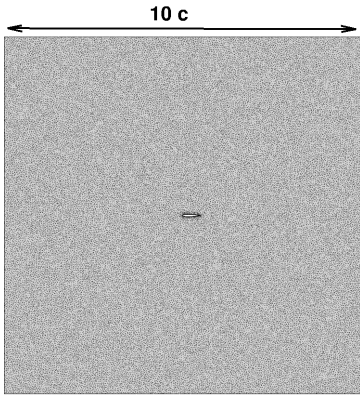
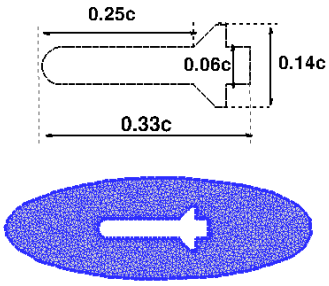


Figure 7: Reference starting point for tank trajectory.

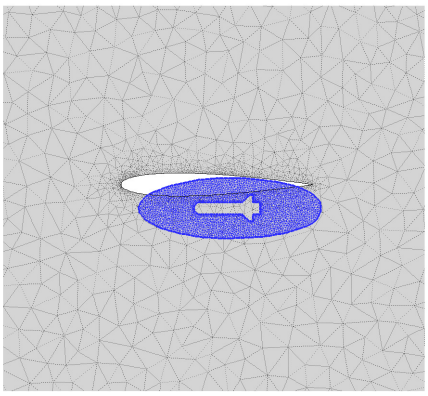
$$\begin{cases} y = y_0 - 0.02 \cdot \Delta t \\ x = x_0 - 0.02 \cdot \Delta t \end{cases} \tag{2}$$



(a) Background grid (32733 nodes, 64637 cells)



(b) Body mesh (1975 nodes, 3625 cells)



(c) Tank location at t=0

Figure 8: Tank fall: grids, tank geometry and initial position.

The time-dependent evolution of the Mach flow-field during the tank fall is reported in Fig. 9, which also shows the computational grid used in each frame. As in the previous test-case, the proposed technique is capable of handling the insertion of the grid around the tank within the background mesh at each time step and to evaluate the flow field during the store fall.

4. Conclusion and future work

In this paper we presented some recent developments of a mesh generation algorithm[3] that allows us to numerically simulate the flow-field that surrounds moving bodies experiencing large relative motion. The proposed technique is characterized by a localized re-meshing and a reduced use of solution interpolation, so as to keep the mesh generation cost and interpolation error as low as possible. It is worth noting that the proposed approach avoids meshes overlapping, which commonly occur in most popular overset methods. Moreover, interpolation is limited to those grid points previously removed for inserting the mobile grid in the computational domain.
Sample applications include: a space launcher stage separation and the tank fall from a wing.

Future work will focus on the application of the proposed mesh generation technique to cases involving viscous flows where boundary layer grids are present. Indeed, the current version of the presented technique does not allow to preserve the boundary layer grids of the background meshes when the mobile mesh approaches a wetted surface. Figure 10 provides an example of this shortcoming: the left frame (Fig. 10a) shows the body-fitted grid around the moving object, which moves inside the computational domain shown in Fig. 10b). A fixed body, which for sake of clarity is a square, is contained within the domain: the presence of a boundary layer grid on all sides of the square can be clearly seen. When the moving body approaches the square, the outer boundary of the moving grid removes the boundary layer cells, which is obviously undesirable. We are currently working to eliminate this important drawback in the current version of the algorithm. In the next papers an updated version of the mesh generation algorithm able to compute viscous flows will presented.

DOI: 10.13009/EUCASS2023-037

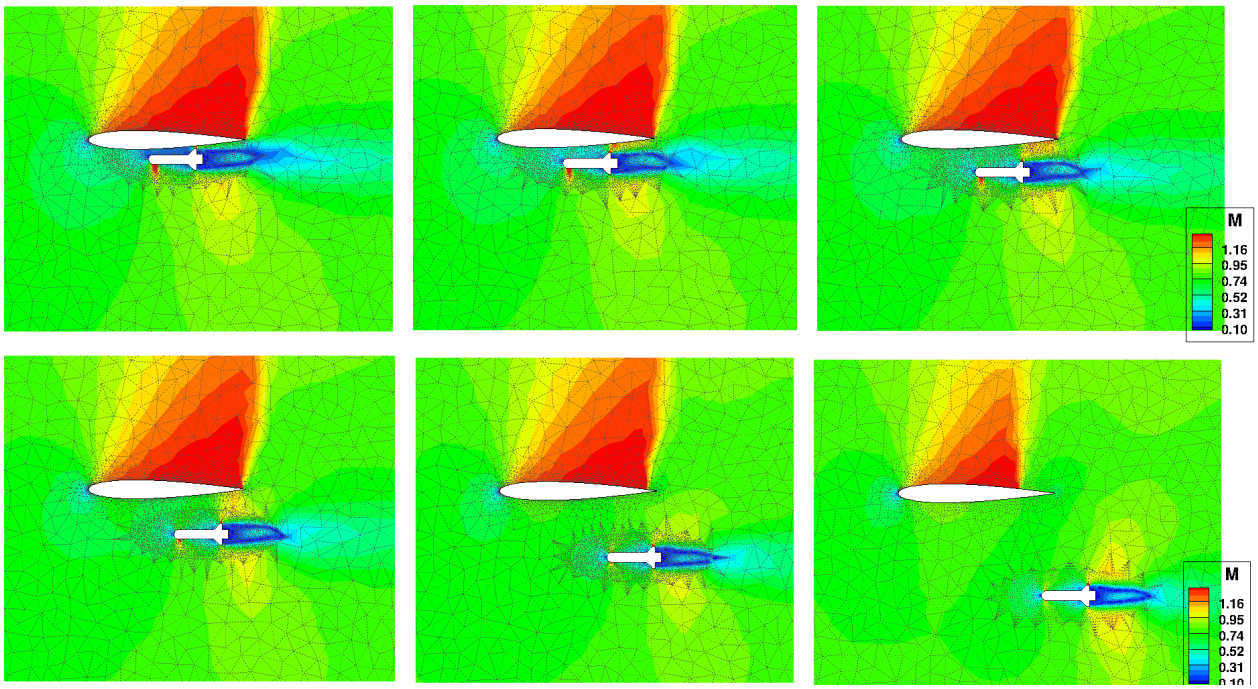
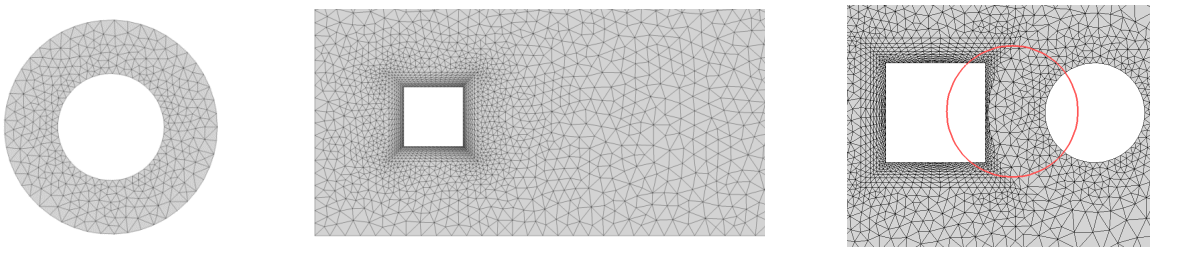


Figure 9: Tank fall: computational grids and Mach flow-field at different time instants.



(a) Mobile mesh (b) Background mesh (c) Resulting mesh generated by MuMs

Figure 10: Example of errors in mesh generation process due to the presence of boundary layer grids.

5. Acknowledgments

This research is jointly funded by Sapienza University and the Italian Space Agency - Agenzia Spaziale Italiana (ASI) as part of the research project N.2019-4-HH.0 0 CUP: F86C17000080005 carried out under a framework agreement between the Parties

References

[1] J Benek, J Steger, and F Carroll Dougherty. A flexible grid embedding technique with application to the euler equations. In *6th computational fluid dynamics conference Danvers*, page 1944, 1983.

[2] Aldo Bonfiglioli. Fluctuation splitting schemes for the compressible and incompressible euler and navier-stokes equations. *International Journal of Computational Fluid Dynamics*, 14(1):21–39, 2000.

[3] Aldo Bonfiglioli and Renato Paciorri. A local, un-structured, re-meshing technique capable of handling large body-motion in rotating machinery. *Energy Procedia*, 82:209–214, 2015.

[4] Sukumar Chakravarthy, Oshin Peroomian, Uriel Goldberg, and Sampath Palaniswamy. The cfd++ computational fluid dynamics software suite. In *AIAA and SAE, 1998 World Aviation Conference*, page 5564, 1998.

[5] Dominic DJ Chandar. On overset interpolation strategies and conservation on unstructured grids in openfoam. *Computer Physics Communications*, 239:72–83, 2019.

DOI: 10.13009/EUCASS2023-037

[6] Eric Haines. Point in polygon strategies. *Graphics Gems*, 4:24–46, 1994.

[7] Hrvoje Jasak and Zeljko Tukovic. Automatic mesh motion for the unstructured finite volume method. *Transactions of FAMENA*, 30(2):1–20, 2006.

[8] Sebastien Lemaire, Guilherme Vaz, Menno Deij-van Rijswijk, and Stephen R Turnock. On the accuracy, robustness, and performance of high order interpolation schemes for the overset method on unstructured grids. *International Journal for Numerical Methods in Fluids*, 94(2):152–187, 2022.

[9] Sebastien Lemaire, Guilherme Vaz, Menno Deij van Rijswijk, and Stephen R. Turnock. Influence of Interpolation Scheme on the Accuracy of Overset Method for Computing Rudder-Propeller Interaction. *Journal of Verification, Validation and Uncertainty Quantification*, 8(1), 01 2023. 011002.

[10] Robert Meakin. On the spatial and temporal accuracy of overset grid methods for moving body problems. In *12th Applied Aerodynamics Conference*, page 1925, 1994.

[11] Jonathan Richard Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Applied Computational Geometry Towards Geometric Engineering: FCRC'96 Workshop, WACG'96 Philadelphia, PA, May 27–28, 1996 Selected Papers*, pages 203–222. Springer, 2005.