Implementation of immersed boundaries through volume penalization in the industrial aeronautical solver CODA

Jonatan Núñez*, David Huergo†, Suyash Shrestha‡, Esteban Ferrer§, and Eusebio Valero¶
Universidad Politécnica de Madrid, 28040 Madrid, Spain

Diego Lodares‖, Juan Guerra**, and Juan Florenciano††
Airbus Defence and Space S.A.U, 28906 Getafe, Spain

In this work, we present the implementation and validation of the immersed boundary volume penalization methods in the CFD solver CODA. Our goal is the modelling and simulation of turbulent fluid flowsin complex 3D aerodynamic configuration through the numerical solution of the Reynolds-Averaged Navier–Stokes equations using the Spalart–Allmaras turbulent model. To do that, an efficient immerse boundary method, together with an efficient preprocessing tool for the construction of unstructured hexahedral meshes with adaptive mesh refinement around immersed geometries has been developed. We report several numerical computations, including subsonic flow past a NACA0012 airfoil, subsonic flow past an MDA30P30N multi-element airfoil and subsonic flow around the NASA high-lift CRM aircraft. These simulations have been performed in the CODA solver with a second-order finite volume scheme as spatial discretization and an implicit backward-Euler scheme based on the matrix-free GMRES block-Jacobi iterative method. The reported numerical simulations are in very good agreement with their corresponding experimental data.

I. Introduction

In the ever-evolving landscape of computational fluid dynamics (CFD) and industrial simulations, the accurate representation of complex geometries and their interaction with fluid flow remains a critical challenge. The ability to model and analyze intricate structures in a computationally efficient manner is paramount, as it directly influences the design, performance, and optimization of various industrial systems. This challenge is particularly relevant in the context of the industrial solver CODA [1], where the interaction between fluid flow and complex geometries plays a central role.

CODA is a parallel software framework for multidisciplinary analysis and optimization of aircraft and helicopters based on advanced and accurate numerical methods [2, 3]. The solver is being developed as part of a collaboration between the French National Aerospace Research Center (ONERA), the German Aerospace Center (DLR) and Airbus. CODA provides a robust, scalable and computational efficient integrated design process for aerodynamics and structural analysis. To perform an efficient analysis and optimization of aircraft on state-of-the-art HPC systems, the Navier–Stokes and the Reynolds-averaged Navier–Stokes (RANS) equations are solved for high Reynolds-number flow on unstructured grids with second-order finite-volume and higher-order discontinuous-Galerkin discretizations.

The implementation of immersed boundaries through volume penalization has emerged as a promising technique to address the difficulty to generate body fitted meshes for complex geometries and simulate moving geometries. Immersed

*Research Scientist, Department of Applied Mathematics, School of Aeronautics; jonatan.nunez@upm.es (Corresponding Author).
†PhD Student, Department of Applied Mathematics, School of Aeronautics.
‡PhD Student, Department of Applied Mathematics, School of Aeronautics.
§Professor, Department of Applied Mathematics, School of Aeronautics.
¶Professor, Department of Applied Mathematics, School of Aeronautics.
‖Engineer, Flight Physics Department.
**Engineer, Flight Physics Department.
††Engineer, Flight Physics Department.

boundaries allow for the inclusion of detailed and intricate geometries within the computational domain, even in cases where traditional structured grids might be impractical or prohibitively expensive to generate. In the realm of CFD simulations, the term "immersed boundary" refers to the representation of complex structures, such as solid objects or porous media, as part of the fluid computational domain, without the need for a grid that conforms to their intricate shapes. This not only enhances the capability to model real-world industrial scenarios, but also significantly improves computational efficiency.

The primary objective of this research is to provide an in-depth exploration of the methodology, mathematical formulations, and computational considerations involved in the implementation of immersed boundaries through volume penalization in the solver CODA. By doing so, this study aims to facilitate a deeper understanding of this technique and its potential applications in aeronautics. It is our hope that the insights and findings presented in this paper will contribute to the advancement of CFD simulations, offering engineers and researchers a valuable tool for the accurate modeling of complex industrial systems and the optimization of their performance.

This paper is organized as follows: In section II we summarize the main numerical schemes implemented in CODA for simulating turbulent fluid flow with immersed boundary methods. In section III we discuss the preprocessing and postprocessing tools developed for generating refined meshes around immersed bodies and the analysis of simulation data. Next, in section IV we present several numerical computations done with CODA with immersed boundary volume penalization methods: subsonic flow past a NACA0012 airfoil, subsonic flow past an MDA30P30N multi-element airfoil and subsonic flow around the NASA high-lift CRM aircraft. Finally, in section V, a summary of this work is presented.

II. Numerical Methods

CODA is a CFD solver, which integrates an extensive set of algorithms for solving partial differential equations in a multi-physics context. In this work, we focus on the the implementation of an immersed boundary methodology for the modelling of turbulent fluid flows by the well-known Reynolds-Averaged Navier–Stokes equations. In our formulation, we have selected the Spalart–Allmaras turbulent model that we briefly reproduce in appendix A for completeness of the paper. Regarding the spatial and time discretization employed in CODA, we summarize the key ingredients in the next section.

A. Spatial and Time Discretization

The spatial discretization schemes used in CODA to solve the Navier–Stokes equations and the Reynolds-Averaged Navier–Stokes equations are the finite volume method [4], the modal discontinuous Galerkin method [5], and the discontinuous Galerkin spectral element method [6, 7]. The partial differential equations of interest can be written as a hyperbolic-parabolic system of conservation laws in differential form (see appendix A for more details)

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \boldsymbol{f}(\boldsymbol{u}, \nabla \boldsymbol{u}) = \boldsymbol{s}\left(\boldsymbol{u}\right), \tag{1}$$

where $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, t)$ is the vector of conserved quantities, and $\boldsymbol{f} = \boldsymbol{f}(\boldsymbol{u}, \nabla \boldsymbol{u})$ is the tensor of physical fluxes, which has two terms, the advective flux and the diffusive term, namely

$$\boldsymbol{f}(\boldsymbol{u}, \nabla \boldsymbol{u}) = \boldsymbol{f}^A(\boldsymbol{u}) - \boldsymbol{f}^D(\boldsymbol{u}, \nabla \boldsymbol{u}), \tag{2}$$

and $\boldsymbol{s} = \boldsymbol{s}\left(\boldsymbol{u}\right)$ represents the source terms. The conservation laws can also be written in integral form as follows

$$\frac{\partial}{\partial t} \int_\Omega \boldsymbol{u} \, \mathrm{d}\Omega + \oint_{\partial\Omega} \left(\boldsymbol{f}^A(\boldsymbol{u}) - \boldsymbol{f}^D(\boldsymbol{u}, \nabla \boldsymbol{u}) \right) \cdot \boldsymbol{n} \, \mathrm{d}\sigma = \int_\Omega \boldsymbol{s} \, \mathrm{d}\Omega. \tag{3}$$

Here, Ω is the spatial domain and $\boldsymbol{n}$ is the normal vector on the surface $\partial\Omega$ enclosing the spatial domain. In this work we are interested only in the finite volume schemes. A short description of these methods is outlined; for the

discontinuous Galerkin schemes, see [8]. In the finite volume framework, the computational domain is divided into a set of non-overlapping polyhedral control volumes or cells, that is $\Omega = \bigcup_i^N \Omega_i$, and the integral form of the equations is discretized for each control volume Ω_i. Defining the mean value of the function $\boldsymbol{u}$ in the cell Ω_i by

$$\boldsymbol{u}_i := \frac{1}{|\Omega_i|} \int_{\Omega_i} \boldsymbol{u}\,(\boldsymbol{x}, t)\ \mathrm{d}\Omega, \tag{4}$$

we get the so-called semi-discrete formulation

$$\frac{\mathrm{d}\boldsymbol{u}_i}{\mathrm{d}t} := -\frac{1}{|\Omega_i|}\left\{\sum_l \left[\boldsymbol{f}^A \cdot \hat{\boldsymbol{n}}\right]^* \sigma_l - \sum_l \left[\boldsymbol{f}^D \cdot \hat{\boldsymbol{n}}\right]^* \sigma_l - \boldsymbol{s}\,|\Omega_i|\right\}. \tag{5}$$

The surface integral has been approximated by a sum of the fluxes through the faces $\partial\Omega_l$ of the cell Ω_i. The σ_l stands for the area of the face $\partial\Omega_l$, and $|\Omega_i|$ is the volume of the cell Ω_i.

The finite volume discretization implemented in CODA is second-order accurate. The advective fluxes are approximated with the Roe schemes with an entropy fix. In this scheme, the left and right states at the cell interfaces are piecewise linearly reconstructed, and therefore gradient-based. More details on how are computed the element and face gradients can be found in [1, 9, 10].

Regarding the time discretization, in CODA are available explicit, implicit and implicit-explicit integrators. In this work we have used an implicit backward-Euler scheme based on preconditioned matrix-free GMRES. Preconditioners include incomplete LU and line-inversion [11].

B. Immersed Boundary Methods

Immersed Boundary Methods are numerical techniques for simulating fluid-structure interaction problems involving complex geometries, providing an alternative to numerical methods based on body-fitted meshes. Immersed Boundary Methods were first proposed by Peskin [12] to simulate blood flow in the human heart. These methods provide a flexible framework for modeling fluid flow over complicated immersed bodies on grids that do not conform to the surface of the body by treating these as immersed boundaries within the computational domain. In this way, the Immersed Boundary Methods overcome the challenges of generating high-quality meshes around complex geometries, which can be time-consuming and computationally demanding, while accurately resolving complex flows using simple grids. The geometric components are discretized using a separate representation, such as a surface or volume mesh, which is then immersed within the computational grid. Immersed boundary conditions are applied to take into account the geometric components lying in the flow.

Because the meshes used in the immersed boundary method are, in principle, non-conforming with the geometry of the obstacle, a very high resolution of this background mesh is required to obtain reliable numerical solutions. The employment of very fine meshes in the whole computational domain makes unpractical the use of the immersed boundary methods for industrial applications, like the study of the aerodynamics of aircrafts. In fact, very fine meshes are only necessary in specific places in the computational domain, like around the immersed geometry and the wake regions. Fine meshes far away from the immersed bodies are unnecessary. For these reasons, mesh refinement algorithms are used to refine the mesh only in the regions where it is actually essential for accurate computations.

1. Immersed Boundary Volume Penalization Method

In this family of immersed boundary volume penalization methods, the flow equations are solved on the whole computational domain, while a source term is introduced to represent the body as a porous medium with very low permeability [13–16]. This method is based on the idea of penalizing the integration points which are located inside the immersed body using source terms, instead of using a boundary condition over the surface of the body, as it is the case when using conforming meshes. In this way, a simple Cartesian mesh can be used to solve the Navier–Stokes equations over arbitrary geometries, reducing the time required for the mesh generation.

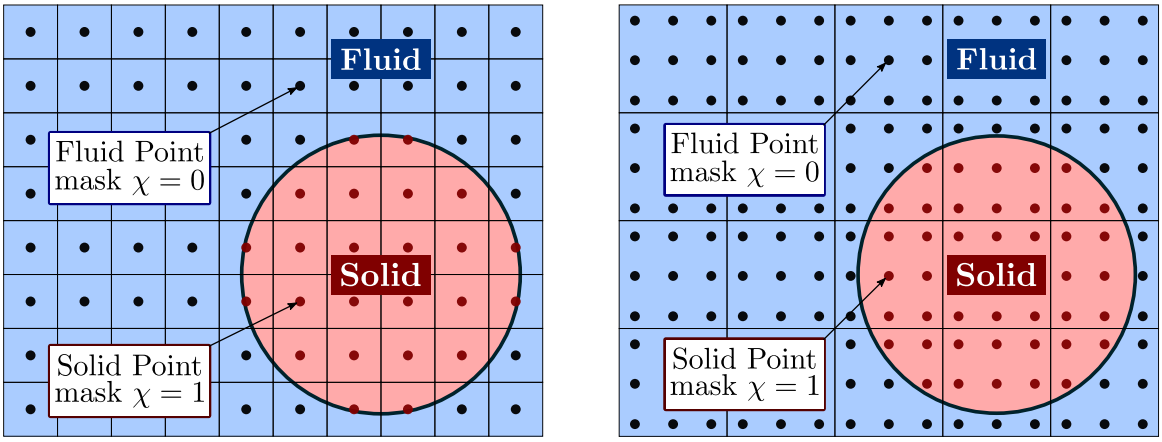

Fig. 1 Schematic diagram of the background mesh for the immersed boundary method as it used along with finite volume schemes (left) and discontinuous Galerkin spectral element methods (right).

In the IBVP method, a mask function is required to discriminate between those points that are inside or outside the immersed body, as it is represented in figure 1. Then, source terms are applied to modify the behavior of the flow field inside the immersed body. Given the governing equations for a compressible viscous fluid:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \boldsymbol{f}(\boldsymbol{u}, \nabla \boldsymbol{u}) = \boldsymbol{s}(\boldsymbol{u}) \tag{6}$$

where $\boldsymbol{s}(\boldsymbol{u})$ is the IBM source term. This source term is written as

$$\boldsymbol{s}(\boldsymbol{u}; \chi, \eta) = \frac{\chi}{\eta} \begin{pmatrix} 0 \\ \rho(\boldsymbol{v} - \boldsymbol{v}_s) \\ \frac{1}{2}\rho\left(\boldsymbol{v} \cdot \boldsymbol{v} - \boldsymbol{v}_s \cdot \boldsymbol{v}_s\right) \end{pmatrix}, \tag{7}$$

where $\boldsymbol{v}_s$ is the velocity of the moving geometry (note that for static object $\boldsymbol{v}_s = 0$ andχ represents the mask function and distinguishes between the fluid region Ω_f and the solid region Ω_s:

$$\chi(\boldsymbol{x}, t) = \begin{cases} 1, & \text{if } \boldsymbol{x} \in \Omega_s, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

and $0 < \eta \ll 1$ is the penalization parameter. If a RANS Spalart–Allmaras model is used, the eddy viscosity, $\tilde{\nu}$, is set to zero inside the body [17]:

$$\boldsymbol{s}_{SA}(\boldsymbol{u}; \chi, \eta) = -\frac{\chi}{\eta}\tilde{\nu}. \tag{9}$$

Furthermore, a wall model has been developed to improve the accuracy of RANS simulations. This model selects a set of points near the immersed body where the wall function will be applied, which are called wall points. For each wall point, the normal vector to the body's surface is computed and a new point is defined, called image point. The image points will provide the required information to the wall function to update the state vector at the wall points. However, image points (in general) are not integration points of the computational mesh. Therefore, the state vector at the image points has to be interpolated from a set of interpolation points. These interpolation points, which are computed through a kd-tree algorithm, are the N-nearest integration points to each image point. An overview of this approach is represented schematically in figure 2. The distance which is used to discriminate between the regions where

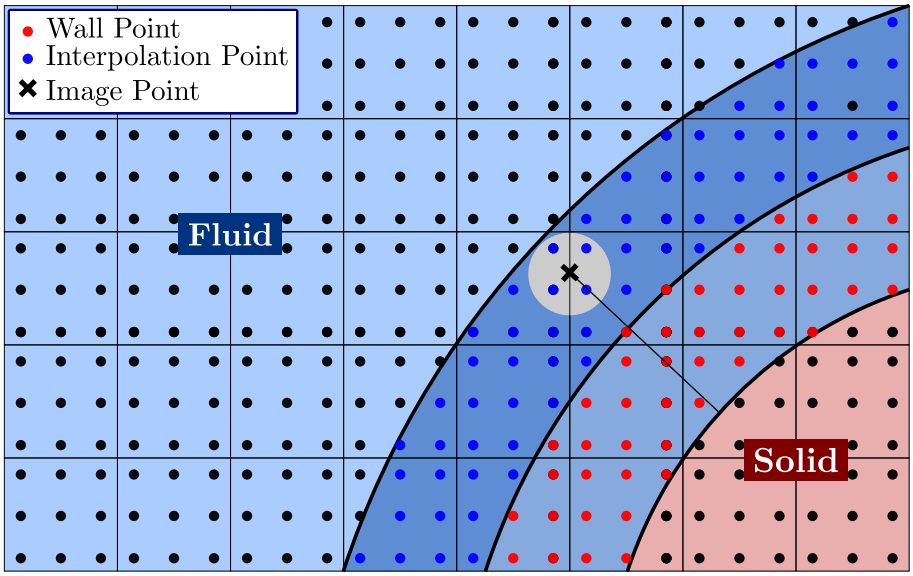

Fig. 2 Schematic definition of wall points, image points and interpolation points.

the wall points and the image points live, is defined as the length of the smallest edge of the smallest cell of the mesh.

To interpolate the state vector at each image point, a Radial Basis Function (RBF) interpolator with a Gaussian kernel is used. Then, a wall function is used to compute the friction velocity at each image point [18]:

$$v_{||}^{+}(y^{+}) = \frac{1}{\kappa}\log\left(1+\kappa y^{+}\right) + \left(C - \frac{1}{\kappa}\log\left(\kappa\right)\right)\left(1 - \exp\left(-\frac{y^{+}}{11}\right) - \frac{y^{+}}{11}\exp\left(-\frac{y^{+}}{3}\right)\right), \tag{10}$$

and

$$v_{\mathrm{IP},t} = v_{\tau} v_{||}^{+}\left(y_{\mathrm{IP}}^{+}\right), \quad \text{with} \quad y_{\mathrm{IP}}^{+} = \frac{v_{\tau} d_{\mathrm{IP}}}{\nu_{\mathrm{IP}}}, \tag{11}$$

where the subscript IP refers to each image point and the subscript t refers to the tangential component of the velocity vector.

Finally, the resulting friction velocity v_{τ} is considered to be constant along the normal direction and, hence, it can be used at the wall points (subscript WP). Taking into account this information, the velocity, the eddy viscosity and the non-dimensional wall distance can be obtained at the wall points:

$$y_{\mathrm{WP}}^{+} = \frac{v_{\tau} d_{\mathrm{WP}}}{\nu_{\mathrm{WP}}}, \tag{12}$$

$$v_{\mathrm{WP},t} = v_{\tau} v^{+}\left(y_{\mathrm{WP}}^{+}\right), \tag{13}$$

$$v_{\mathrm{WP},n} = \frac{d_{\mathrm{WP}}}{d_{\mathrm{IP}}} v_{\mathrm{IP},n}, \tag{14}$$

$$\tilde{\nu}_{\mathrm{WP}} = \kappa v_{\tau} d_{\mathrm{WP}}, \tag{15}$$

which can be solved including an additional condition of parallelism:

$$v_{\mathrm{WP},t} \parallel v_{\mathrm{IP},t} \tag{16}$$

The subscript n refers to the normal direction of the velocity vector, $\kappa = 0.41$ is the von Kármán constant and d is the distance from a point to the immersed body. The new velocity and turbulent viscosity at the wall points are considered

by including additional source terms for the momentum, energy and turbulent viscosity equations. Finally, this wall model as implemented in CODA along with the IBVP scheme has proved to improve the results of the standard RANS simulation if the y^+ at the wall points has a value (approximately) of 60 or less.

2. IBVP Workflow in CODA

The implemented IBVP workflow in CODA consists in the following steps:

Mesh preprocessing In the mesh preprocessing stage, an octree-based Cartesian mesh with refinement around the immersed body is created. This task is done by an external library. The details of how this mesh is generated are given in section III.A.

IBVP preprocessing The immersed bodies are imported by the function **immersedBody.ImmersedBodyGroup** from the **FSImmersedBody** plugin. This plugin contains utilities to place solid bodies inside a CFD mesh, mark points inside them, calculate distances, and generate integration points and weights over them.

CFD simulation CODA loads the octree mesh and calls the preprocessing functions for generating the IBVP mask, and export it with the function **ExportIBVPMaskNSToFSMeshOfExportedIntegrationPoints** if the Navier–Stokes equations are used, and with the function **ExportIBVPMaskRANSToFSMeshOfExportedIntegrationPoints** if the RANS equations are employed. Both functions are part of **immersedBody.ImmersedBodyGroup** and then performs the CFD simulation.

IBVP postprocessing The function **Postprocessing.ComputeLiftAndDrag** computes the lift and drag coefficients. More details are discussed in section III.B.

III. Preprocessing and Postprocessing

A. Preprocessing

Immersed boundary methods use a background mesh for solving a fluid-structure interaction problem. In 3D applications, this background mesh is typically a simple Cartesian mesh made of hexahedral elements. However, it can be also unstructured and made of hybrid elements (hexahedra, prisms, tetrahedra, etc.), or even a body-fitted mesh. In this section we focus on the construction of a Cartesian background mesh built out of hexahedra. This kind of mesh is assembled in an unstructured fashion by the preprocessing tool.

1. Background Mesh Generation

The preprocessing tool has two ways for generating the initial background mesh: the first way consists in creating a new Cartesian mesh from scratch, and the second way consists in importing a mesh generated by the software GMSH (strictly speaking, the preprocessing tool is not generating a mesh but importing an already generated one by an external tool). The preprocessing tool is only capable of handling meshes made of hexahedral elements. Next, we will describe the tasks the preprocessing tool does for constructing an unstructured and adaptively refined Cartesian mesh.

Loading Parameters for Background Mesh Generation

The first task to be performed is to load from file the parameters of the mesh to be constructed. Depending on the type of mesh to be generated (built-in or imported mesh), different parameters are read-in.

For the built-in mesh type, the preprocessing tool requires some parameters that specify the type of built-in box (standard box, curved box, box deformation function, type of target geometry, mesh stretching factors in each spatial direction, etc.). The number of elements in each direction of the Cartesian box, the degree of the mesh (for high-order curved meshes), and the kind of curved boundary are also required.

For the imported mesh type, the filename of the GMSH file is required. Additional parameters are necessary for setting up the name and location of boundary conditions and the output format of the mesh file (formats supported: HDF5, GMSH, and TECPLOT).

Creation of the Background Mesh

Once the mesh parameters are loaded, the preprocessing tool proceeds to create the background mesh, either a built-in or an imported mesh. See Figure 5 for a built-in and an imported mesh generated by the preprocessing tool. The typical workflow of the preprocessing tool is as follows:

- Creation of interpolation matrices between different polynomial bases.
- Creation of a reference box.
- Creation of a curved box from the reference box.
- Application of the stretching functions on each spatial direction, if required.
- Application of mesh deformation functions, if required.
- Application of mesh rotation matrices, if required.
- Storing the final mesh in arrays.
- Numbering of nodes, elements, elements faces and boundary faces.
- Creation of the main data structure of the mesh: the list of elements with their corresponding information (nodes ID, elements ID, local faces definition, boundary conditions). This elements list is a dynamical data structure, namely a linked list. Taking into account that the number of elements increases dynamically according to the desired target mesh size around the immersed geometry, this data structure makes possible a straightforward application of the mesh refinement algorithm.

2. Mesh Refinement around Immersed Geometries

Once the background mesh is generated and stored in the element list data structure, the preprocessing tool will refine this mesh provided the geometry of the body in STL format.

Loading Parameters for the Mesh Refinement

From the same parameter file used in the generation of the background mesh, the preprocessing tool will load all the information necessary for the mesh refinement task: the maximum level of refinement around the immersed geometry, the number and location of special regions that need to be refined, and their corresponding levels of refinement, the type of refinement (either isotropic or anisotropic), and the filenames of the STL files used to describe the geometry of the immersed bodies. After reading this information, the preprocessing tool will perform the mesh refinement around the immersed geometry.

Importing Geometries from STL Files

The immersed geometries are described by STL files. These files contain an unstructured triangulated surface (triangles unit vectors and the coordinates of their corresponding vertices), as they are generated by a CAD software. The preprocessing tool is responsible of loading the tessellated surface of the immersed body for its further employment in the refinement stage.

Mesh Refinement around Immersed Geometries

The preprocessing tool has the capability of refining the background hexahedral mesh in special regions within the computational domain (for instance, wake regions) and also around the triangulated surface of the immersed geometry. The preprocessing tool will follow the next steps for refining the background mesh:

- Flagging the elements that require refinement. For the refinement of special regions in the computational domain, the flagging procedure marks all hexahedral cells inside the region coordinates provided in the parameter file. For the refinement of the hexahedral elements intersecting the triangulated surface of the immersed body, an efficient algorithm for the triangle-box overlap is implemented. Before the mesh refinement takes place, this triangle-box overlap is tested for all triangles of the surface triangulation, and if such overlap takes place, the involved elements store the overlapped triangles. During the mesh refinement loop (refinement level $l > 0$), this triangle-box overlap

is tested only for those hexahedra storing overlapped triangles.

- Flagging the elements for balancing. The FSDM, and therefore CODA can handle hanging nodes for different face element types (triangular and quadrilateral faces). This hanging nodes capability is restricted in FSDM to elements with a 2:1 refinement level ratio. This means that for two neighboring elements with different refinement levels, only a difference of level of at most 1 between the neighboring cells is allowed. The preprocessing tool is aware of this restriction and therefore an algorithm for preserving the 2:1 balancing has been implemented.
- A feature implemented in the preprocessing tool is the refinement of elements around the immersed body up to certain number of neighboring cells far away from the truly overlapped elements with the geometry. This allow us to have a thick layer of small elements around the immersed body, which is useful in the simulation of flow problems with the RANS equations and the employment of wall functions.

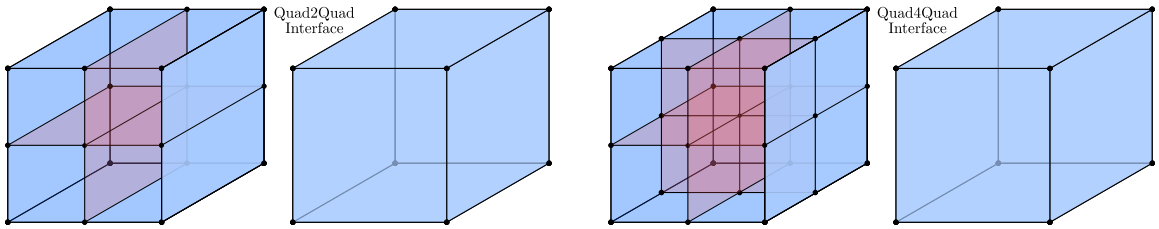

Fig. 3 Hanging nodes distribution in a Quad2Quad interface (left) and Quad4Quad interface (right).

- Refinement of hexahedral elements. All flagged cells are splitted into 4 or 8 new children cells (for anisotropic or isotropic mesh refinement, respectively). Hanging nodes can appear and the relation between the face nodes of neighboring elements with different level of refinement has to be computed and exported in such a way that FSDM can handle them. For the hexahedral meshes used within the preprocessing tool and when hanging nodes are present, the Quad2Quad and Quad4Quad interfaces are computed (see figure 3).
- Storing the final mesh in arrays. In these arrays are stored the nodes, elements, conforming faces and non-conforming faces.
- Exporting the mesh arrays to HDF5 format, following the FSDM guidelines for the HDF5 format.

B. Postprocessing

The postprocessing for the IBVP methodology is based on the computation of the lift and drag of an immersed body, known the geometry (as an STL file) and the converged flow field. Two different approaches are considered: integration over the STL and a Voronoi integration.

Integration over the Original Immersed Body Tessellation

This approach takes advantage of the information within the STL file to perform the integration. First, for each triangle of the tessellation, the fluid variables are interpolated at the barycenter, at the vertices and at the middle point of each edge of the triangle. The interpolation can be performed using different approaches: Radial Basis Functions (RBF), inverse distance or linear interpolation. In general, a simple linear interpolation has shown a good behavior and robustness.

Then the viscous stress tensor (including the pressure) $\tilde{\boldsymbol{\tau}} = \boldsymbol{\tau} - p\boldsymbol{I}$ is computed on those points. Finally, the integration over each triangle is performed as follows:

$$\boldsymbol{F}_t = \frac{S}{60}\left(27\tilde{\boldsymbol{\tau}}_{\text{bary}} + 3\sum_{i=1}^{3}\tilde{\boldsymbol{\tau}}_{\text{vert},i} + 8\sum_{i=1}^{3}\tilde{\boldsymbol{\tau}}_{\text{mid},i}\right). \tag{17}$$

In this case, the integrated force on the triangle t, F_t is computed as a weighted average of the stress tensor (including

the pressure), $\tilde{\tau}$, projected over the normal vector of the local triangle. The value of $\tilde{\tau}_{\text{bary}}$, $\tilde{\tau}_{\text{vert},i}$, and $\tilde{\tau}_{\text{mid},i}$ correspond to the projected force at the barycenter, at the vertex i and at the middle point of the side i respectively.

Integration over a Modified Immersed Body Tessellation

This approach provides an alternative to compute the lift and drag when the STL is too coarse (and the integration over the STL shows a bad quality) or too fine (and the computational cost has to be reduced). In these cases, a new set of points over the surface of the immersed body are defined to perform the integration.

First, a region around the body is defined and every integration point of the mesh within that region is projected over the body's surface. Then, an algorithm is used to compute the weights for each point. Once the points over the surface (and their weights) are known, the fluid variables are interpolated on those points. Finally, the stress tensor (including the pressure) is computed following the same equations shown above and the projected force is integrated as follows:

$$F = \sum_{i=0}^{\text{nPoints}} \tilde{\tau}_i \omega_i. \tag{18}$$

In this case, the global force is computed as the dot product between the projected force at each point, $\tilde{\tau}$, and the weights, ω.

IV. Numerical Computations

To prove the validity and performance of this approach, we present in this section several numerical simulations of different fluid flow problems of increasing dificulty. These tests are the subsonic flow around a NACA0012 airfoil at different angles of attack, the subsonic flow around the MDA30P30N multi-element airfoil, and the more challenging NASA High-Lift Common Research Model (CRM-HL) aircraft (Figure 4-top-left, top-right and bottom respectively). Experimental results of most of those problems are reported in the literature and thus will serve as validation of our methodology. For all simulations, the RANS equations were used for modeling the fluid flow, and the numerical methods employed by CODA were, for the spatial discretization, the second order finite volume method coupled with the immersed boundary volume penalization. A linearized implicit Euler scheme is employed for time discretization. All simulations were executed using 600 cores in the Magerit Supercomputer at Supercomputing and Visualization Center of Madrid (CeSViMa) and 3840 cores in the Marenostrum4 Supercomputer at the Barcelona Supercomputing Center.

A. Flow around a NACA0012 airfoil

The computational domain for the fluid flow around a NACA0012 airfoil is the box with dimensions $[-20, +20] \times [-20, +20] \times [0, +1]$. The span of the domain is sufficient to avoid domain confinement effects and wave reflections from the domain boundaries, which could lead to a significant error when computing the lift and the drag force around the airfoil. The extent of the computational domain corresponds to $40c$, where c is the chord length and is equal to 1. The domain is discretized initially with $n_x n_y n_z = 1600$ hexahedral elements, where $n_x = 40$, $n_y = 40$, and $n_z = 1$ are the number of elements in x, y, and z direction, respectively. This initial mesh is further refined around the immersed geometry and also in the wake region. The grid spacing obeys $h = 2^{-l}$, where l is the level of refinement. Sufficient grid resolution around the airfoil is crucial to obtain accurate results for the lift and drag coefficients. The mesh around the immersed geometry is refined up to level of refinement $l = 12$, we get a mesh with around 2.4×10^6 elements. The wake region is resolved with a refined subregion with refinement level $l = 8$. This region is approximately 20 airfoil chords downstream of the lead edge of the airfoil. The characteristics of the meshes used for the mesh convergence study are shown in table 1; ranging from a very coarse mesh up to a very fine one. The refinement level $l_{\max}$ listed in the table is the highest level of refinement around the immersed body. The immersed geometry is located in the center of the computational domain, with its cross-section on plane z, and this corresponds to the NACA0012 profile with chord length $c = 1$. The triangulation of the surface geometry is made of 18 524 triangles. The geometry is shown in figure 4,

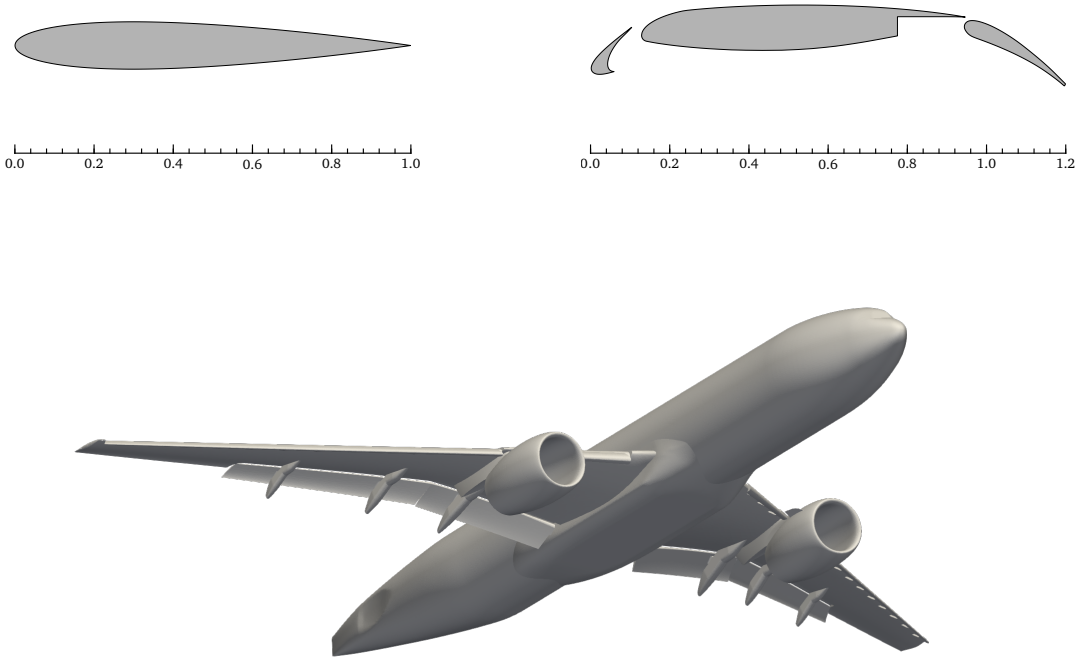

Fig. 4 Geometries used for the numerical simulations: the NACA0012 airfoil (top, left), the MDA30P30N multi-element airfoil (top, right), and the NASA High-Lift CRM Wing-Body-Flap-Slat-Nacelle-Pylon (bottom).

and in figure 5 is depicted the mesh around the immersed geometry.

1. Subsonic flow at $M_\infty = 0.15$ and $\alpha = 0°$ and $\alpha = 10°$

The first test corresponds to the subsonic flow around an NACA0012 airfoil. This problem has become a classic test case for RANS solvers due to the simple geometry and large amount of available numerical and experimental data [19–21]. It is used primarily for the analysis of turbulence models, testing their convergence properties and their effect on the accuracy. We perform the simulations for angles of attack $\alpha = 0°$ and $\alpha = 10°$. The flow satisfies the following conditions: a gas with adiabatic index $\gamma = 1.4$ and Reynolds number $\mathrm{Re} = 6 \times 10^6$ flows with a freestream Mach number $M_\infty = 0.15$. The non-dimensional density is set to $\rho = 1$, and the non-dimensional pressure to $p = 1$. On the boundaries of the computational domain were set the following boundary conditions: left face is set to inflow, the right face to outflow, the front and back faces were set to periodic, and the top and bottom to far field. The inflow pressure assumes the value of the stagnation pressure and the outflow pressure to the static pressure.

In the figure 6 are depicted the contour plots of the velocity magnitude for the flow past the NACA0012 profile at angle of attack $\alpha = 0°$. The mesh employed in the simulation is the very fine mesh with element's size around the body $h_{\mathrm{body}} = 2.44 \times 10^{-4}$, element's size in the wake region $h_{\mathrm{wake}} = 3.91 \times 10^{-3}$. The Spallart–Allmaras model has been activated. We observe the flow field is symmetric as the fluid past the airfoil. As the fluid flows over the blunt trailing edge, a minor amount of flow separation occurs, but it reattaches shortly after it passes the trailing edge and recovers the mean velocity in the distant far field.

Table 1 Characteristics of the meshes for the flow around a NACA0012 airfoil.

Name	$l_{\max}$	h_{wake}	h_{body}	*nElems*
Very Coarse	8	3.91×10^{-3}	3.91×10^{-3}	7.783×10^{5}
Coarse	9	3.91×10^{-3}	1.95×10^{-3}	8.126×10^{5}
Medium	10	3.91×10^{-3}	9.77×10^{-4}	9.115×10^{5}
Fine	11	3.91×10^{-3}	4.88×10^{-4}	1.236×10^{6}
Very Fine	12	3.91×10^{-3}	2.44×10^{-4}	2.391×10^{6}

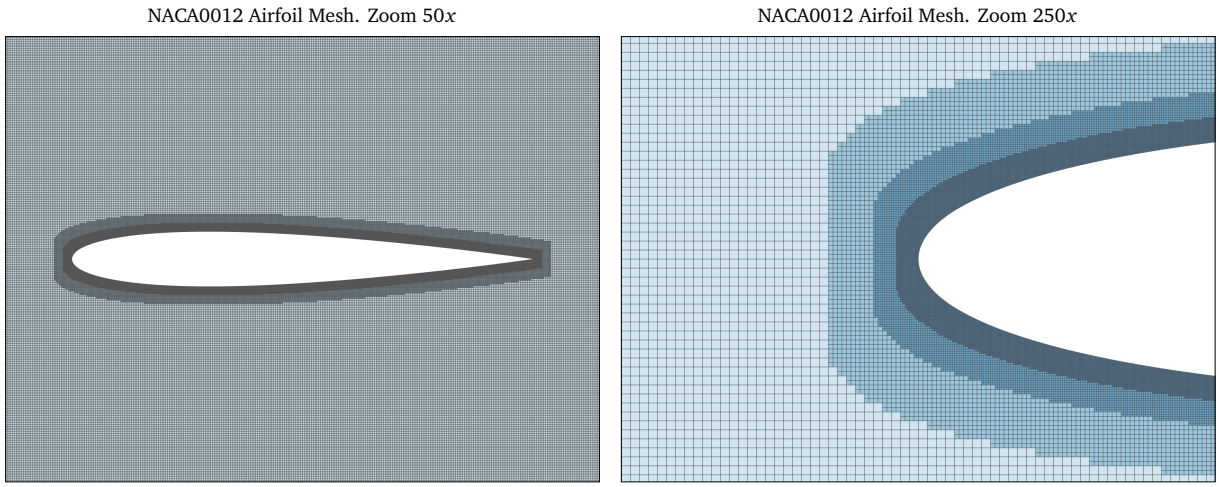

Fig. 5 Mesh refinement around a NACA0012 airfoil (left) and close-up view of the mesh (right).

In the figure 7 are depicted the contour plots of the velocity magnitude for the flow past the NACA0012 profile at angle of attack $\alpha = 10°$. The mesh employed is the same as for angle of attack $\alpha = 0°$. The Spallart–Allmaras model has been activated. We observe that the stagnation point is shifted downward (relative to the airfoil), reflective of the shifted angle of attack. The fluid velocity above the airfoil is clearly different than that below the airfoil, showing a low-velocity region toward the upper side of the trailing edge, which means flow separation.

Now we analyze the pressure coefficient plots. The figure 8 and figure 9 show pressure coefficient C_p over the airfoil surface for all considered meshes and also for the cases with wall model deactivated and activated. Experimental data are also plotted and they were obtained from [19]. For an angle of attack $\alpha = 0°$, and for all considered meshes, the pressure coefficient curve is very close to the experimental data, but some unphysical oscillations are clearly perceptible in all simulations, both with the wall model switched off and switched off. These oscillations tend to diminish as the mesh becomes finer. For an angle of attack $\alpha = 10°$, the pressure coefficient curves for the very coarse, coarse and medium meshes show a large discrepancy regarding the experimental data, especially on the leading edge upper surface. Only the fine and very fine meshes present a good agreement with the experiment. The activation of the wall model helps to reduce this disagreement between the numerical results and experimental data for the coarse meshes.

Values of the lift and drag are shown in table 2 for angle of attack $\alpha = 0°$ and in table 3 for angle of attack $\alpha = 10°$. For the simulation with angle of attack $\alpha = 10°$, the lift coefficient C_L increases monotonically as the grid is refined, getting closer to the experimental value, while the drag coefficient C_D appears to decrease monotonically and also it is approaching its numerical value to the experimental one. This behavior occurs for the simulations with the wall model deactivated and activated. For the simulation with angle of attack $\alpha = 0°$, we do not observe monotonicity in the evolution of the lift coefficient C_L and the drag coefficient C_D as the mesh is refined.

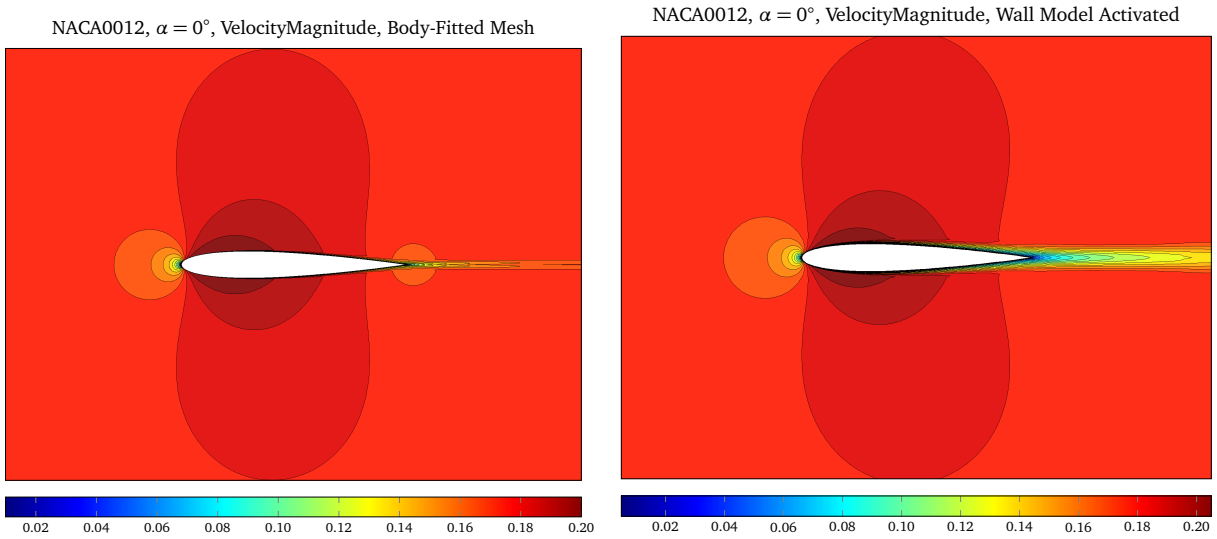

Fig. 6 Subsonic flow around the NACA0012 airfoil: contour plots of the velocity magnitude for the simulations with angle-of-attack $\alpha = 0°$ (top) and $\alpha = 10°$ (bottom). In both computations, the RANS equations were solved with the wall model activated.

Table 2 Subsonic flow around the NACA0012 airfoil: Lift and drag coefficients for angle of attack $\alpha = 0°$.

	Wall Model Deactivated		Wall Model Activated	
Mesh	C_L	C_D	C_L	C_D
Very Coarse	-8.72×10^{-5}	1.94×10^{-2}	4.85×10^{-6}	1.48×10^{-2}
Coarse	-2.06×10^{-5}	2.16×10^{-2}	8.27×10^{-6}	1.34×10^{-2}
Medium	-6.18×10^{-5}	2.34×10^{-2}	1.37×10^{-5}	1.51×10^{-2}
Fine	-2.02×10^{-5}	2.26×10^{-2}	1.69×10^{-4}	1.68×10^{-2}
Very Fine	-8.21×10^{-5}	1.99×10^{-2}	2.65×10^{-4}	1.93×10^{-2}
Body-Fitted	—	—	-9.43×10^{-6}	8.33×10^{-3}
Experimental	—	—	-7.24×10^{-3}	8.09×10^{-3}

B. Subsonic flow around an MDA30P30N multi-element airfoil

The third test in the benchmarking is a three-element configuration, specifically the McDonnell Douglas 30P-30N landing configuration (MDA30P30N). Many experimental and computational studies have been performed for the flow past multi-element airfoils in the last decades [22–32]. Accurate prediction of the flow over multi-element airfoils during high-lift operations can improve the performance and the safety factor of aircrafts. The flow around multi-element airfoils is complex and it is well known to be dominated by different flow mechanisms at different operating conditions, making rather difficult to accurately predict high-lift flow fields. The geometry of the configuration is depicted in figure 4. The leading-edge slat and the trailing-edge flap have a deflection angle of 30°. This airfoil has been tested extensively at NASA Langley Low Turbulence Pressure Tunnel at various Reynolds and Mach numbers and also has been numerically simulated using a wide range of numerical techniques for solving the Navier–Stokes equations along with different turbulence models.

The computational domain is similar to the employed in the NACA0012 case, that is, a box with dimensions $[-20, +20] \times [-20, +20] \times [0, +1]$. The domain is discretized initially with $n_x n_y n_z = 1600$ hexahedral elements, where $n_x = 40$, $n_y = 40$, and $n_z = 1$. The mesh is then refined around the immersed geometry and in the wake region. The mesh around the immersed geometry is refined up to level of refinement $l = 12$, and the wake region is resolved with

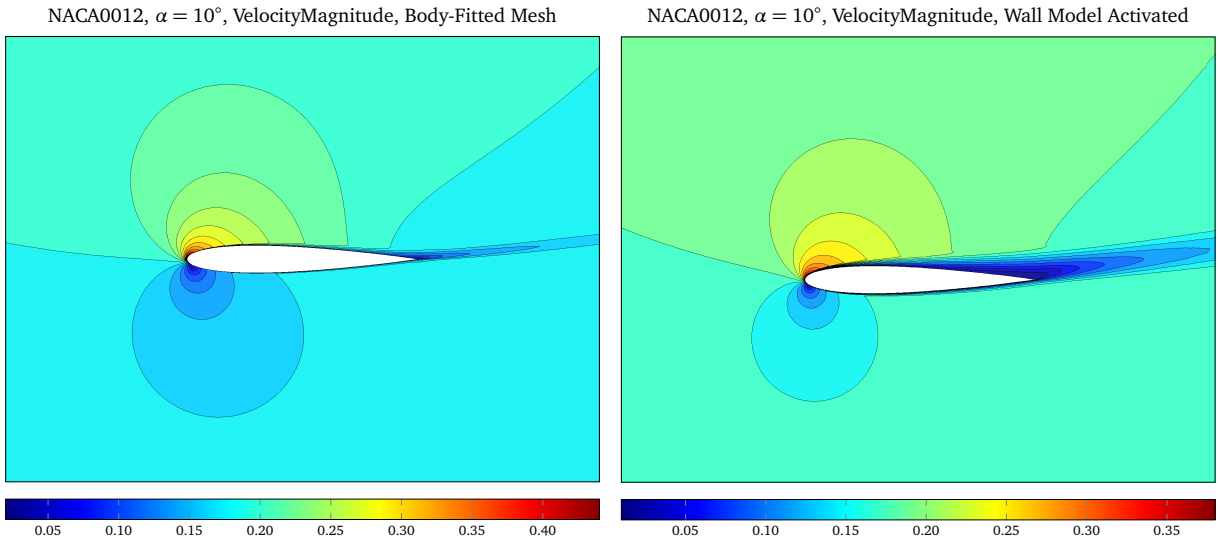

Fig. 7 Subsonic flow around the NACA0012 airfoil: contour plots of the velocity magnitude for the simulations with angle-of-attack $\alpha = 0°$ (top) and $\alpha = 10°$ (bottom). In both computations, the RANS equations were solved with the wall model activated.

Table 3 Subsonic flow around the NACA0012 airfoil: Lift and drag coefficients for angle of attack $\alpha = 10°$.

	Wall Model Deactivated		Wall Model Activated	
Mesh	C_L	C_D	C_L	C_D
Very Coarse	4.56×10^{-1}	2.12×10^{-1}	4.30×10^{-1}	1.91×10^{-1}
Coarse	4.72×10^{-1}	2.07×10^{-1}	5.05×10^{-1}	2.05×10^{-1}
Medium	5.41×10^{-1}	2.09×10^{-1}	6.56×10^{-1}	2.03×10^{-1}
Fine	8.19×10^{-1}	1.95×10^{-1}	7.75×10^{-1}	1.88×10^{-1}
Very Fine	8.45×10^{-1}	1.91×10^{-1}	8.00×10^{-1}	1.90×10^{-1}
Body-Fitted	—	—	1.090	1.23×10^{-2}
Experimental	—	—	1.057	1.19×10^{-2}

a refined subregion with refinement level $l = 8$. This region is approximately 20 airfoil chords downstream of the lead edge of the airfoil. In table 4 are shown the characteristics of the meshes used for the mesh convergence study, ranging from a medium mesh up to a very fine mesh. The refinement level $l_{\max}$ listed in the table is the highest level of refinement around the immersed body. The immersed geometry is located in the center of the computational domain, with its cross-section on plane z, and this corresponds to the MDA30P30N profile with chord length $c = 1.2$. The triangulation of the surface geometry is made of 66 320 triangles. In figure 10 is depicted the mesh around the immersed geometry. The inner elements are not shown.

Regarding the flow conditions, the gas has adiabatic index $\gamma = 1.4$ and it flows with Reynolds number $\mathrm{Re} = 9 \times 10^6$ and freestream Mach number $M = 0.2$. Simulations were run with the airfoil at an angle of attack $\alpha = 8°$. The non-dimensional density is set to $\rho = 1$, and the non-dimensional pressure $p = 1$. On the boundaries of the computational domain were set the following boundary conditions: left face is set to inflow, the right face to outflow, the front and back faces were set to periodic, and the top and bottom to far field. The inflow pressure assumes the value of the stagnation pressure and the outflow pressure to the static pressure.

Contour plots of velocity magnitude and streamtraces from the numerical solution on the very fine mesh are shown

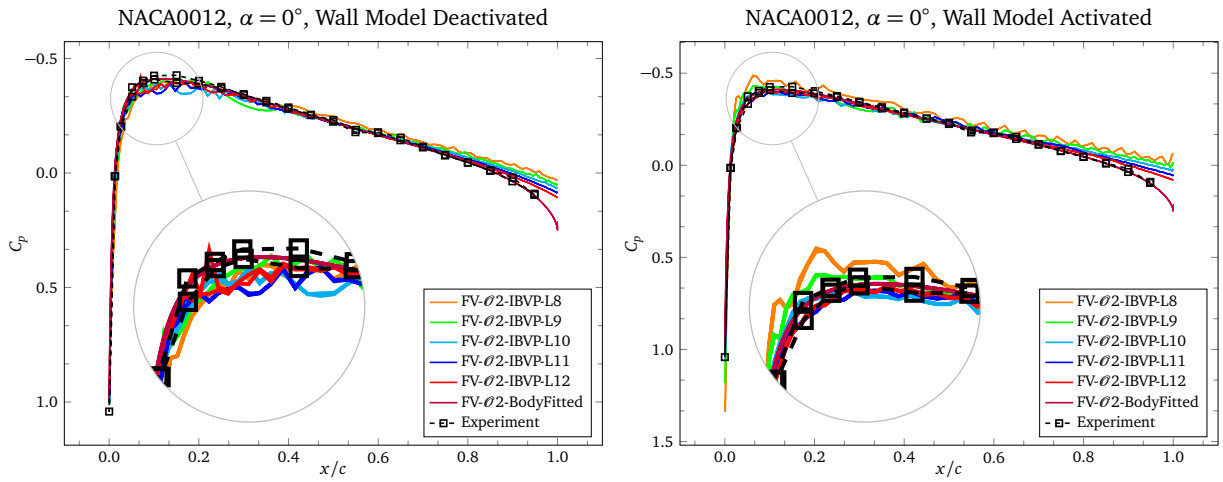

Fig. 8 Subsonic flow around the NACA0012 airfoil: grid convergence study in terms of the pressure coefficient C_p for the subsonic flow over the NACA0012 airfoil surface at angle of attack $\alpha = 0°$ (top) and $\alpha = 10°$ (bottom), and for the cases with wall model deactivated (left) and activated (right).

Table 4 Characteristics of the meshes for the flow around an MDA30P30N multi-element airfoil.

Name	$l_{\max}$	h_{wake}	h_{body}	*nElems*
Medium	10	3.91×10^{-3}	9.77×10^{-4}	1.136×10^{6}
Fine	11	3.91×10^{-3}	4.88×10^{-4}	1.473×10^{6}
Very Fine	12	3.91×10^{-3}	2.44×10^{-4}	2.613×10^{6}

in figure 11 and figure 12. A small separation region is present on the suction side of the upstream side of the trailing edge of the flap. In the figure 13 is presented the grid convergence in terms of the pressure coefficient C_p over the multi-element airfoil surface, with wall model deactivated and activated. Experimental data were taken from [33]. In the main part of the airfoil, the pressure coefficient curve is in good agreement with the experimental data, except at its trailing edge. We can observe a remarkable discrepancy with respect to the experimental data on the upper surface of the slat and the flap.

C. Subsonic flow around NASA high-lift CRM

The last test we consider in this work is the NASA CRM high-lift configuration (CRM-HL), selected from the Fourth AIAA CFD High Lift Prediction Workshop [34]. The NASA Common Research Model has been used mainly in the Drag Prediction Workshop (DPW) and the High Lift Prediction Workshop (HLPW) assess the accuracy of numerical methods in the prediction of aircraft forces and moments. The database from these workshops allow to assess the robustness and effectiveness of state-of-the-art numerical programs and turbulence modeling techniques using Navier–Stokes solvers.

The CRM configuration was initially designed by Boeing and further manufactured and tested by NASA [35]. The NASA CRM consists of a contemporary supercritical transonic wing and a fuselage that is representative of a wide-body commercial transport aircraft. Several experiments of the CRM have been carried out in several research facilities [36–38].

The computational domain used for this simulation is a box with dimensions $[-20, +20] \times [-20, +20] \times [0, +20]$. The domain is discretized initially with $n_x n_y n_z = 32\,000$ hexahedral elements, where $n_x = 40$, $n_y = 40$, and $n_z = 20$. The mesh is then refined around the immersed geometry and in the wake region. The mesh around the immersed

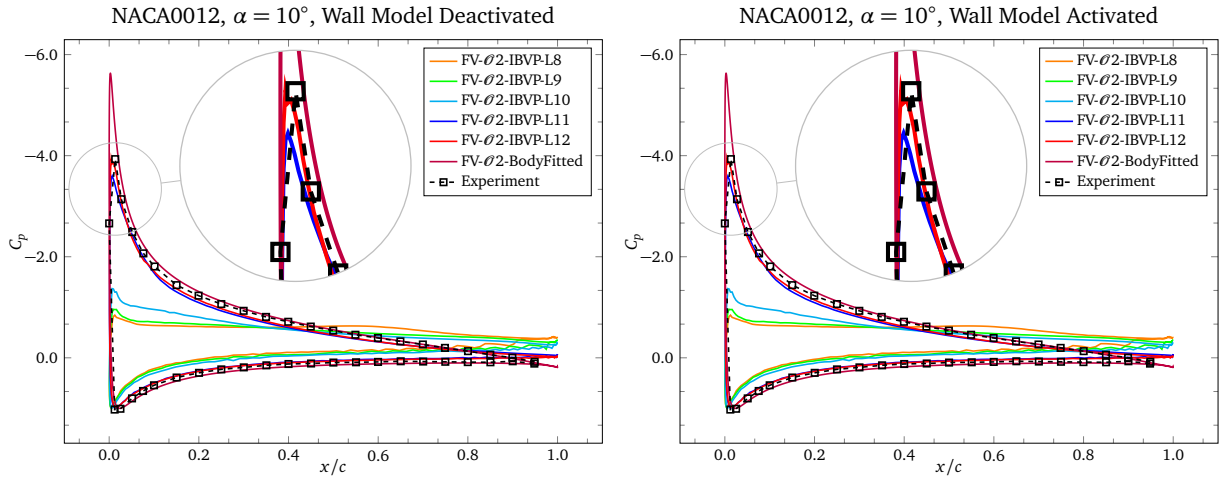

Fig. 9 Subsonic flow around the NACA0012 airfoil: grid convergence study in terms of the pressure coefficient C_p for the subsonic flow over the NACA0012 airfoil surface at angle of attack $\alpha = 0°$ (top) and $\alpha = 10°$ (bottom), and for the cases with wall model deactivated (left) and activated (right).

geometry is refined up to level of refinement $l = 10$, and the wake region is resolved with a refined subregion with refinement level $l = 8$. In table 5 are shown the characteristics of the meshes used for the mesh convergence study, ranging from a very coarse mesh up to a medium mesh. The refinement level $l_{\max}$ listed in the table is the highest level of refinement around the immersed geometry. The geometry of the configuration is depicted in figure 4. The immersed

Table 5 Characteristics of the meshes for the flow around a NASA high-lift CRM Wing-Body-Flap-Slat-Nacelle-Pylon geometry.

Name	$l_{\max}$	h_{wake}	h_{body}	*nElems*
Very Coarse	8	3.91×10^{-3}	3.91×10^{-3}	5.827×10^{6}
Coarse	9	3.91×10^{-3}	1.95×10^{-3}	23.207×10^{6}
Medium	10	3.91×10^{-3}	9.77×10^{-4}	96.520×10^{6}

geometry is located in the center of the computational domain. The triangulation of the surface geometry is made of 74 331 triangles. In figure 10 is depicted the mesh around the immersed geometry.

The Reynolds number based on the mean aerodynamic chord (MAC = 7.005 m) is Re = 5.6 million, and the angle-of-attack is $\alpha = 8°$ with an incoming Mach number of $M_\infty = 0.2$. On the boundaries of the computational domain were set the following boundary conditions: left face is set to inflow, the right face to outflow, the front and back faces were set to periodic, and the top and bottom to far field. The inflow pressure assumes the value of the stagnation pressure and the outflow pressure to the static pressure.

V. Conclusions

The immersed boundary volume penalization method for the Navier–Stokes and the RANS equations has been implemented in the CFD solver CODA. The immersed boundary method is compatible with the finite volume, modal discontinuous Galerkin and discontinuous Galerkin spectral element methods and with the explicit and implicit temporal schemes available in CODA. An efficient preprocessing tool for the construction of unstructured hexahedral meshes with adaptive mesh refinement around immersed geometries has been developed. The octree meshes generated with the tool are imported in CODA and used for simulating complex aerodynamics flow problems. Immersed boundary methods require a very simple meshing process: only the box dimension and number of elements of the Cartesian mesh

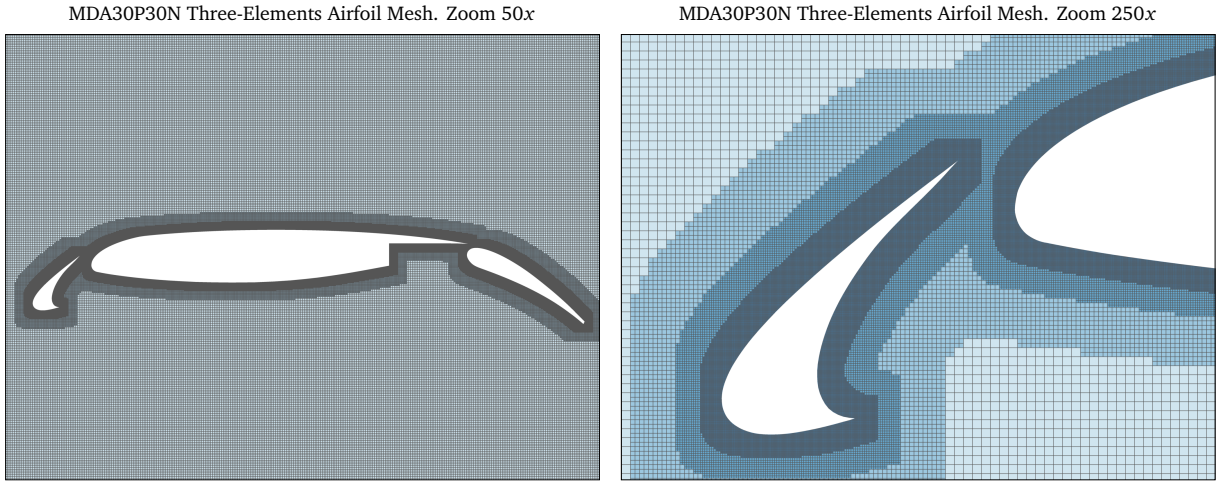

Fig. 10 Mesh refinement around an MDA30P30N airfoil (left) and close-up view of the slat mesh (right).

and the geometry of the body are required as an input. The automatic mesh generation process is able to automatically refine the mesh near the body. Besides, the immersed boundary technology can be used straightforward with an external predefined mesh, without any additional modification.

Several numerical computations were presented and discussed: subsonic flow past a NACA0012 airfoil, subsonic flow past an MDA30P30N multi-element airfoil and subsonic flow around the NASA high-lift CRM aircraft. The computations are in very good agreement with their corresponding experimental data for the pressure coefficient curve, and the drag and lift coefficients when a wall model for the RANS equations is used.

Even though the immersed boundary methodology works within the CFD solver CODA with very good results, a critical disadvantage of these techniques with respect to a body-fitted approach is that the mesh refinement around the immersed geometry leads to a very high number of elements when a high resolution of the boundary layer is desired. Anisotropic mesh refinement reduce the number of elements in comparison with isotropic mesh refinement, but the mesh size is still prohibitive.

As a consequence of these considerations, we will explore also alternative approaches to automatic mesh generation for IBM. In particular, future work will focus on hybrid IBM approaches, in which only geometrical details (control surfaces, ice shapes, etc) will be immersed in a body-fitted, wall-resolved mesh of a clean aerodynamic surface. This approach would benefit from IBM advantages (several configurations of the geometrical details can be tested on the same body-fitted background mesh) without handling meshes of very big sizes.

VI. Acknowledgments

Jonatan Núñez, David Huergo, Esteban Ferrer and Eusebio Valero acknowledge the funding received by the Grant NextSim/AEI/10.13039/501100011033 and H2020, GA-956104. This project has received funding from the Clean Aviation Joint Undertaking under the European Union's Horizon Europe research and innovation programme under Grant Agreement HERA (Hybrid-Electric Regional Architecture) no. 101102007. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or CAJU. Neither the European Union nor the granting authority can be held responsible for them. Esteban Ferrer and Eusebio Valero acknowledge the funding received by the Grant DeepCFD (Project No. PID2022-137899OB-I00) funded by MCIN/AEI/10.13039/501100011033 and by ERDF A way of making Europe. Esteban Ferrer would like to thank the support of the Comunidad de Madrid and Universidad Politécnica de Madrid for the Young Investigators award: APOYO-JOVENES-21-53NYUB-19-RRX1A0. Finally, all authors gratefully acknowledge Universidad Politécnica de Madrid (`www.upm.es`) for providing computing resources on Magerit Supercomputer. The authors also thankfully

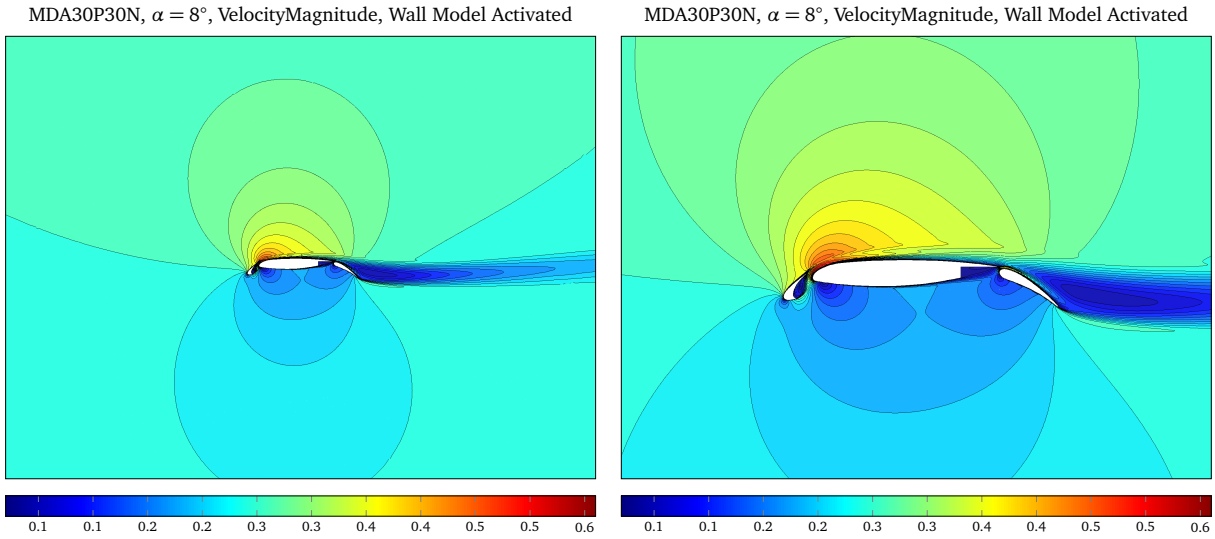

Fig. 11 Subsonic flow around an MDA30P30N multi-element airfoil: contour plots of the velocity magnitude of the slat (left) and the flap (right). RANS equations were solved with the wall model activated.

acknowledge the computer resources at MareNostrum and the technical support provided by Barcelona Supercomputing Center (RES-IM-2022-3-0023).

References

[1] Volpiani, P., Chapelier, J.-B., Schwöppe, A., Jägersküpper, J., and Champagneux, S., "Aircraft simulations using the new CFD software from ONERA, DLR, and Airbus," *Journal of Aircraft*, Vol. 0, No. 0, 2024, pp. 1–13. https://doi.org/10.2514/1.C037506.

[2] Kroll, N., Abu-Zurayk, M., Dimitrov, D., Franz, T., Führer, T., Gerhold, T., Görtz, S., Heinrich, R., Ilic, C., Jepsen, J., Jägersküpper, J., Kruse, M., Krumbein, A., Langer, S., Liu, D., Liepelt, R., Reimer, L., Ritter, M., Schwöppe, A., Scherer, J., Spiering, F., Thormann, R., Togiti, V., Vollmer, D., and Wendisch, J.-H., "DLR project Digital-X: towards virtual aircraft design and flight testing based on high-fidelity methods," *CEAS Aeronautical Journal*, Vol. 7, 2016, pp. 3–27. https://doi.org/10.1007/s13272-015-0179-7.

[3] Jägersküpper, J., and Vollmer, D., "On highly scalable 2-level-parallel unstructured CFD," *Proceedings of the 8th European Congress on Computational Methods in Applied Sciences and Engineering*, 2022. https://doi.org/10.23967/eccomas.2022.208.

[4] Blazek, J., *Computational Fluid Dynamics: Principles and Applications*, 3rd ed., Elsevier Science, Amsterdam, 2015.

[5] Hesthaven, J., and Warburton, T., *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, Springer, Heidelberg, 2008.

[6] Kopriva, D., *Implementing Spectral Methods for Partial Differential Equations*, Springer, Berlin, 2009.

[7] Giraldo, F., *An Introduction to Element-Based Galerkin Methods on Tensor-Product Bases*, Springer, Berlin, 2020.

[8] Basile, F., Chapelier, J.-B., de la Llave Plata, M., Laraufie, R., and Frey, P., "Unstructured h- and hp-adaptive strategies for discontinuous Galerkin methods based on a posteriori error estimation for compressible flows," *Computers & Fluids*, Vol. 233, 2022, p. 105245. https://doi.org/10.1016/j.compfluid.2021.105245.

[9] Schwöppe, A., and Diskin, B., *Accuracy of the cell-centered grid metric in the DLR TAU-code*, Springer Nature, Berlin, 2013, pp. 429–437. https://doi.org/10.1007/978-3-642-35680-3_51.

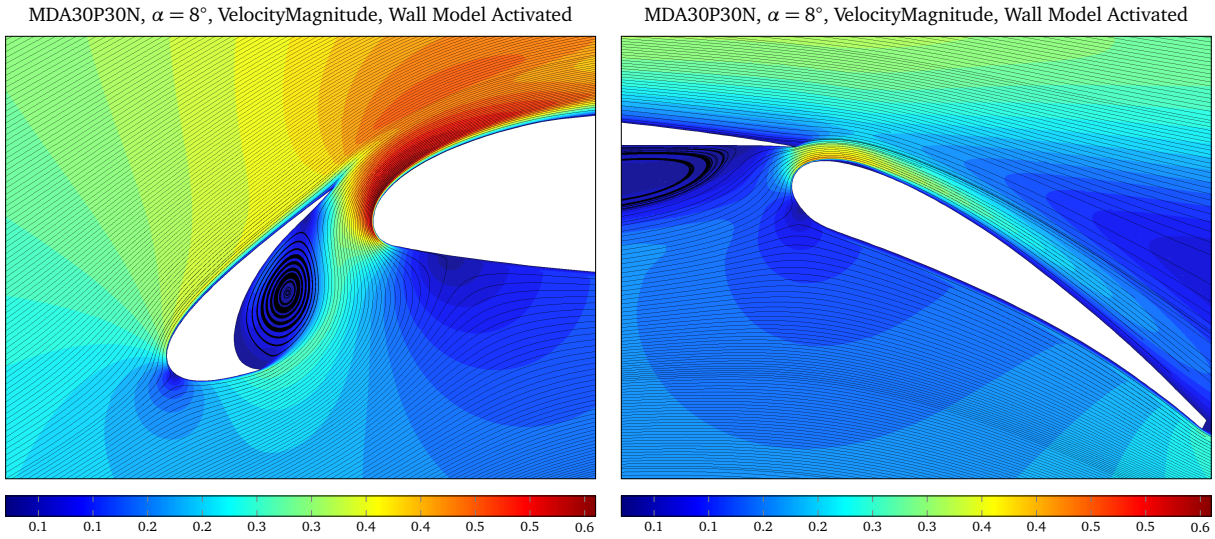

Fig. 12 Subsonic flow around an MDA30P30N multi-element airfoil: streamtraces of the slat (left) and the flap (right). RANS equations were solved with the wall model activated.

[10] Langer, S., Schwöppe, A., and Leicht, T., *Comparison and unification of finite-volume discretization strategies for the unstructured node-centered and cell-centered grid metric in TAU and CODA*, Springer Nature Switzerland, Cham, 2024, pp. 262–272. https://doi.org/10.1007/978-3-031-40482-5_25.

[11] Mohnke, J., and Wagner, M., *A look at performance and scalability of the GPU accelerated sparse linear system solver Spliss*, Springer Nature Switzerland, Cham, 2023, pp. 637–648. https://doi.org/10.1007/978-3-031-39698-4_43.

[12] Peskin, C., "Flow patterns around heart valves: a numerical method," *Journal of Computational Physics*, Vol. 10, 1972, pp. 252–271. https://doi.org/10.1016/0021-9991(72)90065-4.

[13] Kou, J., Joshi, S., Hurtado-de Mendoza, A., Puri, K., Hirsch, C., and Ferrer, E., "Immersed boundary method for high–order flux reconstruction based on volume penalization," *Journal of Computational Physics*, Vol. 448, 2022, p. 110721. https://doi.org/10.1016/j.jcp.2021.110721.

[14] Kou, J., Hurtado-de Mendoza, A., Joshi, S., Le Clainche, S., and Ferrer, E., "Eigensolution analysis of immersed boundary method based on volume penalization: applications to high-order schemes," *Journal of Computational Physics*, Vol. 449, 2022, p. 110817. https://doi.org/10.1016/j.jcp.2021.110817.

[15] Llorente, V., Kou, J., Valero, E., and Ferrer, E., "A modified equation analysis for immersed boundary methods based on volume penalization: applications to linear advection-diffusion and high-order discontinuous Galerkin schemes," *Computers & Fluids*, Vol. 257, 2023, p. 105869. https://doi.org/10.1016/j.compfluid.2023.105869.

[16] Ferrer, E., Rubio, G., Ntoukas, G., Laskowski, W., Mariño, O., Colombo, S., Mateo-Gabín, A., Marbona, H., Manrique de Lara, F., Huergo, D., Manzanero, J., Rueda-Ramírez, A., Kopriva, D., and Valero, E., "HORSES3D: a high order discontinuous Galerkin solver for flow simulations and multiphysics applications." *Computer Physics Communications*, Vol. 287, 2023, p. 108700. https://doi.org/10.1016/j.cpc.2023.108700.

[17] Tamaki, Y., Harada, M., and Imamura, T., "Near-wall modification of Spalart–Allmaras turbulence model for immersed boundary method," *AIAA Journal*, Vol. 55, No. 9, 2017, pp. 3027–3039. https://doi.org/10.2514/1.J055824.

[18] Frère, A., Carton de Wiart, C., Hillewaert, K., Chatelain, P., and Winckelmans, G., "Application of wall-models to discontinuous Galerkin LES," *Physics of Fluids*, Vol. 29, No. 8, 2017, p. 085111. https://doi.org/10.1063/1.4998977.

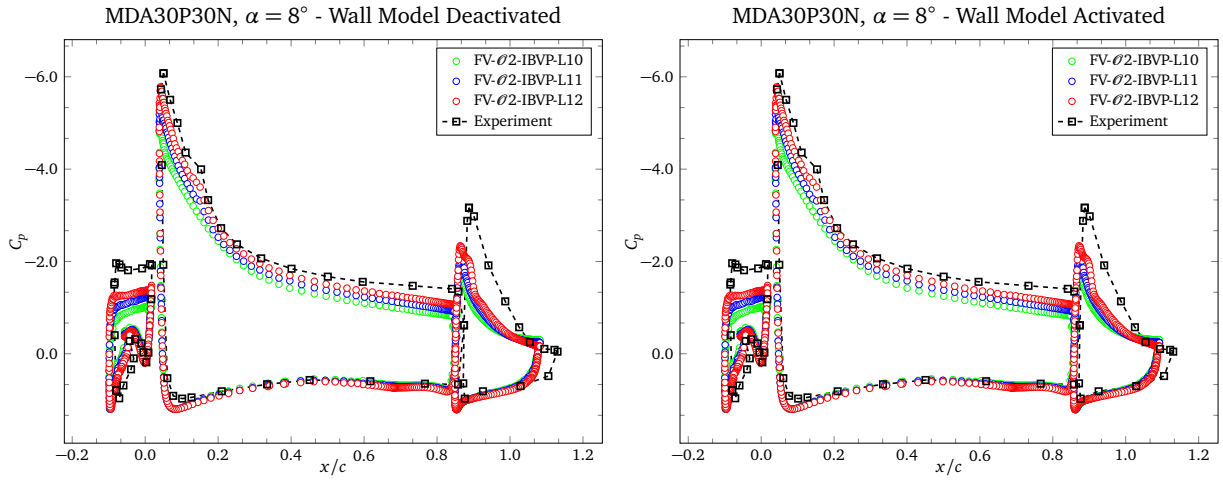

Fig. 13 Subsonic flow around an MDA30P30N multi-element airfoil: grid convergence study in terms of the pressure coefficient C_p for the subsonic flow over the MDA30P30N multi-element airfoil surface at angle of attack $\alpha = 8°$, for the cases with wall model deactivated (left) and activated (right).

[19] Ladson, C., Hill, A., and Johnson, W., "Pressure distributions from high Reynolds number transonic tests of an NACA 0012 airfoil in the Langley 0.3-meter transonic cryogenic tunnel," techreport NASA TM-100526, NASA Langley Research Center, Hampton, Virginia, 1988.

[20] Ladson, C., and Hill, A., "High Reynolds number transonic tests on an NACA 0012 airfoil in the Langley 0.3-meter transonic cryogenic tunnel," techreport NASA TM-100527, NASA Langley Research Center, Hampton, Virginia, 1988.

[21] Ladson, C., "Effects of independent variation of Mach and Reynolds numbers on the low-speed aerodynamic characteristics of the NACA 0012 airfoil section," techreport NASA TM-4074, NASA Langley Research Center, Hampton, Virginia, 1988.

[22] Valarezo, W., Dominik, C., McGhee, R., Goodman, W., and Paschal, K., "Multi-element airfoil optimization for maximum lift at high Reynolds numbers," *Proceedings of the 9th AIAA Applied Aerodynamics Conference*, 1991. https://doi.org/10.2514/6.1991-3332.

[23] Valarezo, W., "High-lift testing at high Reynolds numbers," *Proceedings of the 17th AIAA Aerospace Ground Testing Conference*, 1992. https://doi.org/10.2514/6.1992-3986.

[24] Chin, V., Peters, D., Spaid, F., and McGhee, R., "Flowfield measurements about a multi-element airfoil at high Reynolds numbers," *Proceedings of the 23rd AIAA Fluid Dynamics, Plasma Dynamics, and Lasers Conference*, 1993. https://doi.org/10.2514/6.1993-3137.

[25] Rogers, S., "Progress in high-lift aerodynamic calculations," *Journal of Aircraft*, Vol. 31, No. 6, 1994, pp. 1244–1251. https://doi.org/10.2514/3.46642.

[26] Rogers, S., Menter, F., Durbin, P., and Mansour, N., "A comparison of turbulence models in computing multi-element airfoil flows," *Proceedings of the 32nd AIAA Aerospace Sciences Meeting and Exhibit*, 1994. https://doi.org/10.2514/6.1994-291.

[27] Anderson, W., Bonhaus, D., McGhee, R., and Walker, B., "Navier–Stokes computations and experimental comparisons for multielement airfoil configurations," *Journal of Aircraft*, Vol. 32, No. 6, 1995, pp. 1246–1253. https://doi.org/10.2514/3.46871.

[28] Klausmeyer, S., and Lin, J., "Comparative results from a CFD challenge over a 2D three-element high-lift airfoil," techreport NASA TM-112858, NASA Langley Research Center, Hampton, Virginia, 1997.

[29] Bertelrud, A., "Transition on a three-element high lift configuration at high Reynolds numbers," *Proceedings of the 36th AIAA Aerospace Sciences Meeting and Exhibit*, 1998. https://doi.org/10.2514/6.1998-703.

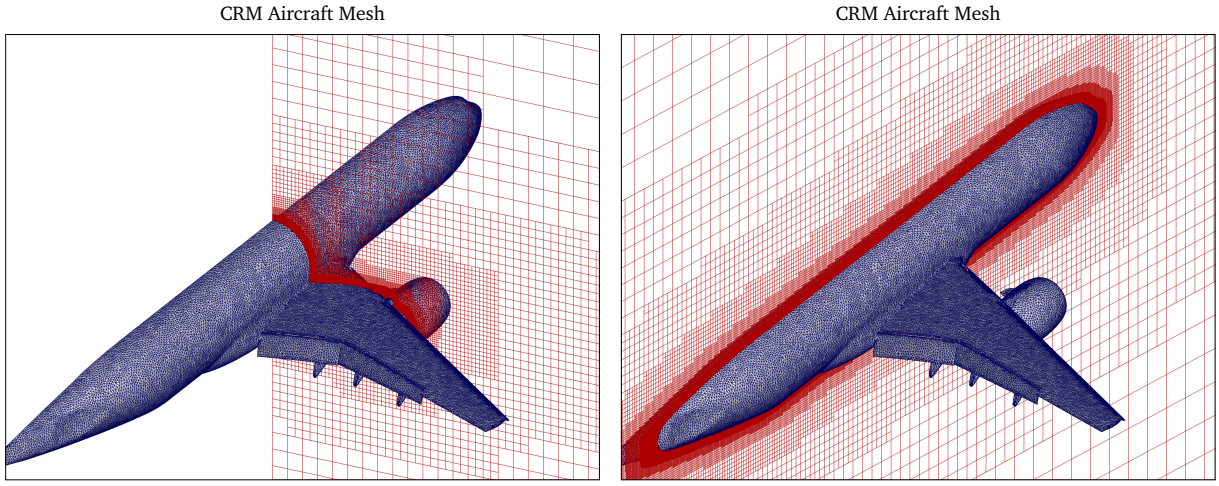

Fig. 14 Mesh refinement around the NASA high-lift CRM Wing-Body-Flap-Slat-Nacelle-Pylon geometry.

[30] Rumsey, C., Gatski, T., Ying, S., and Bertelrud, A., "Prediction of high-lift flows using turbulent closure models," *AIAA Journal*, Vol. 36, No. 5, 1998, pp. 765–774. https://doi.org/10.2514/2.435.

[31] Liou, W., and Liu, F., "Computational modeling for the flow over a multi-element airfoil," *Proceedings of the 17th AIAA Applied Aerodynamics Conference*, 1999. https://doi.org/10.2514/6.1999-3177.

[32] Spaid, F., "High Reynolds number, multielement airfoil flowfield measurements," *Journal of Aircraft*, Vol. 37, No. 3, 2000, pp. 499–507. https://doi.org/10.2514/2.2626.

[33] Murayama, M., Nakakita, K., Yamamoto, K., Ura, H., Ito, Y., and Choudhari, M., "Experimental study on slat noise from 30P30N three-element high-lift airfoil at JAXA Hard-Wall Lowspeed Wind Tunnel," *Proceedings of the 20th AIAA/CEAS Aeroacoustics Conference*, 2014. https://doi.org/10.2514/6.2014-2080.

[34] Ashton, N., Batten, P., Cary, A., and Holst, K., "Summary of the 4th High-Lift Prediction Workshop Hybrid RANS/LES Technology Focus Group," *Journal of Aircraft*, Vol. 61, No. 1, 2024, pp. 86–115. https://doi.org/10.2514/1.C037329.

[35] Vassberg, J., Dehaan, M., Rivers, M., and Wahls, R., "Development of a Common Research Model for applied CFD validation studies," *Proceedings of the 26th AIAA Applied Aerodynamics Conference*, 2008. https://doi.org/10.2514/6.2008-6919.

[36] Rivers, M., and Dittberner, A., "Experimental investigations of the NASA Common Research Model," *Journal of Aircraft*, Vol. 51, No. 4, 2014, pp. 1183–1193. https://doi.org/10.2514/1.C032626.

[37] Ueno, M., Kohzai, M., Koga, S., Kato, H., Nakakita, K., Sudani, N., and Nakamura, Y., "Normalization of wind-tunnel data for NASA Common Research Model," *Journal of Aircraft*, Vol. 52, No. 5, 2015, pp. 1535–1549. https://doi.org/10.2514/1.C032989.

[38] Cartieri, A., Hue, D., Chanzy, Q., and Atinault, O., "Experimental investigations on Common Research Model at ONERA-S1MA–Drag Prediction Workshop numerical results," *Journal of Aircraft*, Vol. 55, No. 4, 2018, pp. 1491–1508. https://doi.org/10.2514/1.C034414.

A. Governing Equations

A. Navier–Stokes Equations

The Navier–Stokes equations can be written as a hyperbolic-parabolic system of conservation laws in the following way

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \boldsymbol{f}(\boldsymbol{u}, \nabla \boldsymbol{u}) = \boldsymbol{0}. \tag{19}$$

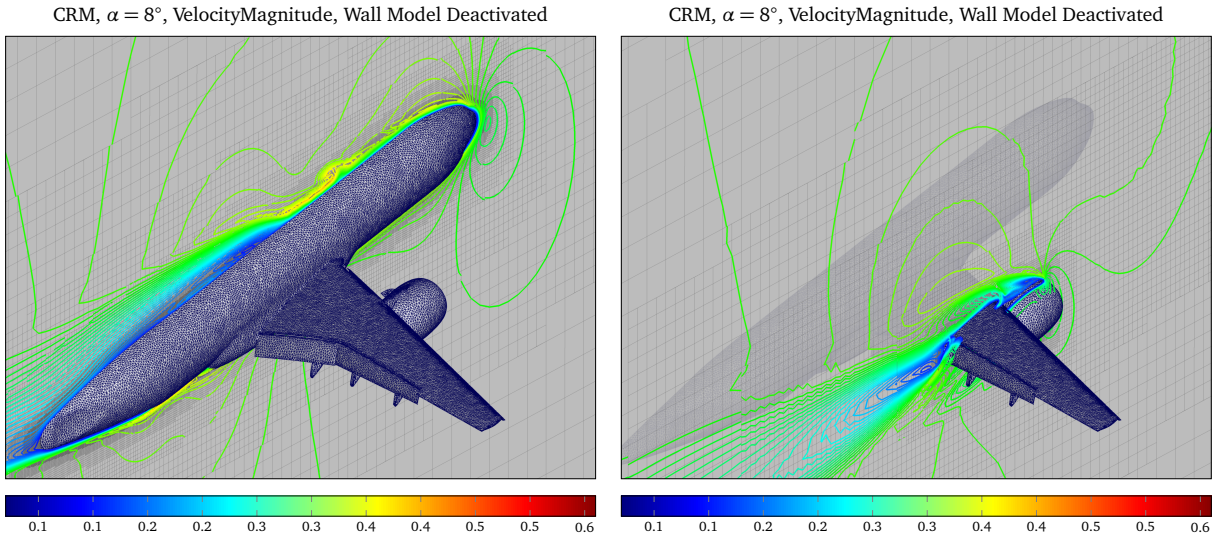

Fig. 15 Subsonic flow around NASA high-lift CRM: contour plots of the velocity magnitude at different positions. In both results, the RANS equations were solved with the wall model deactivated.

The vector of conserved quantities is defined by $\boldsymbol{u}(\boldsymbol{x}, t) = (\rho, \rho\boldsymbol{v}, \rho E)$, where ρ is the mass density, $\boldsymbol{v} = \left(v_x, v_y, v_z\right)$ is the velocity vector, and E is the total energy. The physical flux is defined by

$$\boldsymbol{f}(\boldsymbol{u}, \nabla\boldsymbol{u}) = \boldsymbol{f}^A(\boldsymbol{u}) - \boldsymbol{f}^D(\boldsymbol{u}, \nabla\boldsymbol{u}), \tag{20}$$

with the convective and viscous fluxes given, respectively, by

$$\boldsymbol{f}^A(\boldsymbol{u}) = \begin{pmatrix} \rho\,\boldsymbol{v} \\ \rho\,\boldsymbol{v} \otimes \boldsymbol{v} + p\boldsymbol{I} \\ \boldsymbol{v}\,(\rho E + p) \end{pmatrix}, \quad \boldsymbol{f}^D(\boldsymbol{u}, \nabla\boldsymbol{u}) = \begin{pmatrix} 0 \\ -\boldsymbol{\tau} \\ \boldsymbol{\tau}\cdot\boldsymbol{v} - \boldsymbol{q} \end{pmatrix}. \tag{21}$$

The viscous stresses are described by the stress tensor $\boldsymbol{\tau}$, defined by

$$\boldsymbol{\tau} = 2\mu\boldsymbol{S}^D = \mu\left(\nabla\boldsymbol{v} + (\nabla\boldsymbol{v})^\intercal - \frac{2}{3}(\nabla\cdot\boldsymbol{v})\,\boldsymbol{I}\right), \tag{22}$$

where $\boldsymbol{S}^D$ is the deviatoric component of the strain-rate tensor

$$\boldsymbol{S} = \frac{1}{2}(\nabla\boldsymbol{v} + (\nabla\boldsymbol{v})^\intercal) \tag{23}$$

and

$$\boldsymbol{q} = -k\nabla T, \quad k = \mu\frac{c_p}{\mathrm{Pr}}. \tag{24}$$

The equation of state

$$p = (\gamma - 1)\left(\rho E - \frac{1}{2}\rho v^2\right), \tag{25}$$

where p is the static pressure, γ is the ideal gas index, μ is the dynamic viscosity, T is the temperature, k is the thermal conductivity, and μ denotes the dynamic viscosity coefficient. The kinematic viscosity coefficient is defined by the

formula

$$\nu = \mu/\rho. \tag{26}$$

B. Reynolds-Averaged Navier–Stokes Equations

The Navier–Stokes equations govern the motion of fluids in both laminar and turbulent regimes. Because the large range of spatial and temporal scales present in turbulent flows, solving directly the Navier–Stokes equations is prohibitively expensive. Therefore the Reynolds-averaged Navier–Stokes (RANS) equations are solved instead to model steady turbulent mean flows. The RANS equations couple the mean flow equations with the one-equation turbulence model of Spallart–Allmaras. These equations can be written also as a hyperbolic-parabolic system of conservation laws with source term in the following way

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot \boldsymbol{f}(\boldsymbol{u}, \nabla \boldsymbol{u}) = \boldsymbol{s}(\boldsymbol{u}). \tag{27}$$

Here, $\boldsymbol{u}$ is the vector of time-averaged conservative variables over a given time interval, and has the following components: $\boldsymbol{u}(\boldsymbol{x}, t) = (\rho, \rho \boldsymbol{v}, \rho E, \rho \tilde{\nu})$, where ρ is the time-averaged mass density, $\boldsymbol{v} = \left(v_x, v_y, v_z\right)$ is the time-averaged velocity vector, E is the time-averaged total energy, and $\rho\tilde{\nu}$ is a new conservative variable relating the time-averaged mass density and the eddy viscosity $\tilde{\nu}$. The advective and diffusive components of the physical flux associated to the quantity $\rho\tilde{\nu}$ are given by

$$\boldsymbol{f}^A\left[\rho\tilde{\nu}\right] = \rho\tilde{\nu}\boldsymbol{v}, \quad \boldsymbol{f}^D\left[\rho\tilde{\nu}\right] = \frac{1}{\sigma}\left(\mu + f_{n1}\rho\tilde{\nu}\right)\nabla\tilde{\nu} \tag{28}$$

In the diffusive fluxes, the turbulent stress tensor $\boldsymbol{\tau}_t$ and the turbulent heat fluxes $\boldsymbol{q}_t$ are added respectively to $\boldsymbol{\tau}$ and $\boldsymbol{q}$ appearing in the mean flow equations:

$$\boldsymbol{\tau}_t = 2\mu_t \boldsymbol{S}^D, \quad \boldsymbol{q}_t = -\frac{\mu_t}{\mathrm{Pr}_t} c_p \nabla T \tag{29}$$

where $\mathrm{Pr}_t = 0.9$ is the turbulent Prandtl number and μ_t is the turbulent dynamic viscosity

$$\mu_t = \begin{cases} \rho\tilde{\nu} f_{v1}(\chi) & \text{for } \tilde{\nu} \geq 0, \\ 0 & \text{for } \tilde{\nu} < 0, \end{cases}, \quad f_{v1}(\chi) = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \quad \chi = \frac{\rho\tilde{\nu}}{\mu}. \tag{30}$$

The source terms only have nonzero component in the equation for the turbulent variable $\rho\tilde{\nu}$:

$$\boldsymbol{S}\left[\rho\tilde{\nu}\right] = -\rho(P - D) - \frac{c_{b2}}{\sigma}\rho\nabla\tilde{\nu} + \frac{1}{\sigma}\left(\nu + f_{n1}\tilde{\nu}\right)\nabla\rho \cdot \nabla\tilde{\nu}, \tag{31}$$

where the production and destruction terms, P and D, are defined by:

$$P = \begin{cases} c_{b1}(1 - f_{t2})\tilde{\omega}\tilde{\nu} & \text{for } \tilde{\nu} \geq 0, \\ c_{b1}(1 - c_{t3})\omega\tilde{\nu} & \text{for } \tilde{\nu} < 0, \end{cases}$$

$$D = \begin{cases} \left(c_{w1} f_w - \frac{c_{b1}}{\kappa^2} f_{t2}\right)\left(\frac{\tilde{\nu}}{d}\right)^2 & \text{for } \tilde{\nu} \geq 0, \\ -c_{w1}\left(\frac{\tilde{\nu}}{d}\right)^2 & \text{for } \tilde{\nu} < 0, \end{cases}$$

and

$$f_{n1} = \begin{cases} 1 & \text{for } \tilde{\nu} \geq 0, \\ \frac{c_{n1} + \chi^3}{c_{n1} - \chi^3} & \text{for } \tilde{\nu} < 0, \end{cases}, \quad f_{t2} = c_{t3} \exp\left(-c_{t4}\chi^2\right), \quad f_w = g\left(\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}\right)^{\frac{1}{6}} \tag{32}$$

with

$$g = r + c_{w2}\left(r^6 - r\right), \quad r = \min\left(r_{\lim}, \frac{\tilde{\nu}}{\omega\kappa^2 d^2}\right), \tag{33}$$

and d is the distance to the nearest wall and ω the vorticity magnitude. The modified vorticity magnitude $\tilde{\omega}$ is given by

$$\tilde{\omega} = \begin{cases} \omega + \bar{\omega} & \text{for } \bar{\omega} > -c_{v2}\omega, \\ \omega + \frac{\omega\left(c_{v2}^2 + c_{v3}\bar{\omega}\right)}{(c_{v3} - 2c_{v2})\omega - \bar{\omega}} & \text{for } \bar{\omega} < -c_{v2}\omega, \end{cases} \tag{34}$$

where $\bar{\omega}$ and f_{v2} are given by

$$\bar{\omega} = \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}, \quad f_{v2} = 1 - \frac{\chi}{1 - f_{v1}} \tag{35}$$

For the sake of completeness, we give the values of the constants in the above expressions: $c_{v1} = 7.1$, $\sigma = 2/3$, $c_{b1} = 0.1355$, $c_{b2} = 0.622$, $\kappa = 0.41$, $c_{w2} = 0.3$, $c_{w3} = 2$, $r_{\text{lim}} = 10$, $c_{t3} = 1.2$, $c_{t4} = 0.5$, $c_{v2} = 0.7$, $c_{v3} = 0.9$, $c_{n1} = 16$.