Engineering with Computers (2021) 37:2419–2437
https://doi.org/10.1007/s00366-020-00950-y

ORIGINAL ARTICLE

An immersed boundary method on Cartesian adaptive grids for the simulation of compressible flows around arbitrary geometries

S. Péron[1] · C. Benoit[1] · T. Renaud[1] · I. Mary[1]

Received: 13 June 2019 / Accepted: 10 January 2020 / Published online: 11 February 2020

Abstract
In this article, we present an immersed boundary method for the simulation of compressible flows of complex geometries encountered in aerodynamics. The immersed boundary methods allow the mesh not to conform to obstacles, whose influence is taken into account by modifying the governing equations locally (either by a source term within the equation or by imposing the flow variables or fluxes locally, similarly to a boundary condition). A main feature of the approach which we propose is that it relies on structured Cartesian grids in combination with a dedicated HPC Cartesian solver, taking advantage of their low memory and CPU time requirements but also the automation of the mesh generation and adaptation. Turbulent flow simulations are performed by solving the Reynolds-averaged Navier–Stokes equations or by a Large-Eddy simulation approach, in combination with a wall function at high Reynolds number, to mitigate the cell count resulting from the isotropic nature of Cartesian cells. The objective of this paper is to demonstrate that this automatic workflow is fast and robust and enables to get quantitative aerodynamics results on geometrically complex configurations. Results obtained are in good agreement with classical body-fitted approaches but with a significant reduction of the time of the whole process, that is a day for RANS simulations, including the mesh generation.

Keywords Immersed boundaries · Cartesian adaptive grids · Turbulent flows · Wall law

1 Introduction

The rise of computational fluid dynamics (CFD) in aerospace sciences in the past decades is due to the growth of the computational power in combination with the increase of robustness and accuracy of CFD solvers. Today, Reynolds-averaged Navier–Stokes simulations on body-fitted meshes are commonly performed by the aeronautical industry in the design phase. The geometrical complexity of the configurations has increased too, taking into account for more details, such as track fairings on an aircraft or rotor head components for an helicopter. Consequently, the mesh generation, which requires usually manual interaction and expertise, has become a major bottleneck of the CFD workflow.

This means that efficient tools are required to perform parametric studies and to evaluate quickly the impact of a modification of a shape or some details onto the performances of an aircraft. High-fidelity CFD tools are generally not necessary at this stage; lifting-line tools can be used to get trends quickly, but models are often limited to certain flow assumptions. Low-fidelity CFD (e.g., Euler solutions) could be appropriate but automatic mesh generation is the barrier to override. The immersed boundary methods (IBM) can be seen as a good compromise between the quality of the solution and how quickly it can be obtained. This concept refers initially to the work of Peskin [34, 35], which employed a novel approach many decades ago to simulate biological flows onto Cartesian grids which did not conform to the geometry. The obstacles lying in the flow are taken into account by introducing a forcing term into the momentum equations. Since then, many variants of this approach have been developed, as quoted by Mittal and Iaccarino [29]. A first approach consists in introducing a continuous source term and is well suited for flows with immersed elastic boundaries [34, 37]. In this context, the source term represents the exchange of momentum between the fluid and solid through a law based on the theory of elasticity. However, in the limit of rigid boundaries, this problem is stiff, leading to a lack of stability and accuracy.

✉ S. Péron
stephanie.peron@onera.fr

1 ONERA - Université Paris Saclay, Châtillon 92322, France

Several discrete forcing methods have been developed for flow simulations around solid bodies, among which the ghost-cell direct forcing approach, as developed by Mittal et al. [28], Fadlun et al. [19], and Tseng et al. [45]. The IBM can be used on the whole geometry [32, 45] or locally [30, 47] to capture the potential effects of geometrical details. A similar approach consists in cutting cells that intersect the geometry which has proven efficient and robust for inviscid flow simulations and low Reynolds flows around complex geometries (see [6, 15]).

The use of Cartesian grids with local grid refinement in combination with embedded obstacles (either with immersed boundary or cut-cell methods) seems to be well suited for a high-level of automation and computational efficiency [6, 8, 32]. Although the use of adaptive Cartesian grids around arbitrary immersed obstacles is conceptually attractive, the resolution of high Reynolds number flows requires wall models [7, 10] to restrict the number of points within the boundary layer.

This paper proposes an efficient, fast, and robust immersed boundary method on adaptive structured Cartesian grids to perform CFD simulations of compressible flows. The method relies on a second-order accurate finite-Volume HPC solver dedicated to Cartesian grids, enabling to deal with a wide range of flow regimes, from subsonic to supersonic flows, for steady RANS or LES simulations. Musker's algebraic wall function [31] is applied within the IBM approach on Cartesian grids to solve high-Reynolds number flows. In addition, the immersed boundary approach has been extended to enable several types of immersed boundary conditions. For example, an outlet with imposed pressure or an injection can be modeled locally, or a wall slip immersed boundary condition (which can be used to represent a ground or a wind tunnel) and a wall-modeled condition (for the model in the wind tunnel for instance).

This paper is organized as follows: in Sect. 2, the Ghost Cell Direct Forcing IBM approach used here is described. The way which the different immersed boundary conditions are reconstructed at each iteration is detailed. Section 3 describes how this approach is meaningful when applied on Cartesian adaptive grids: an automatic workflow starting from input surfaces describing immersed boundaries has been developed, in combination with a HPC dedicated Cartesian solver, providing results within a short delay. Section 4 presents two IBM simulations: the first simulation is an RANS simulation of a tripod mounted in a wind tunnel, to illustrate the ability of the approach to deal with different immersed boundary conditions and with a geometrically complex configurations, for which a reduced human effort was required. The second simulation is an unsteady simulation of the flow around a high-lift airfoil for which we focus on aerodynamics results, compared to experimental data and results obtained with body-fitted LES simulation aerodynamics.

2 Description of the immersed boundary method

2.1 Governing equations

The Navier–Stokes equations for a compressible flow can be expressed as follows:

$$\begin{cases} \frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho u_j\right) = 0 \\ \frac{\partial \rho u_i}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho u_i u_j\right) = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\sigma_{ij}\right) \quad i = 1,2,3 \\ \frac{\partial \rho E}{\partial t} + \frac{\partial}{\partial x_j}\left((\rho E + p)u_j\right) = -\frac{\partial}{\partial x_j}\left(Q_j\right) + \frac{\partial}{\partial x_j}\left(\sigma_{ij}u_i\right), \end{cases} \tag{1}$$

where ρ denotes the fluid density, u the velocity vector, p the pressure, ρE the total energy per unit mass, σ the viscous stress tensor, and Q the heat flux vector. In our approach, the system (1) is solved for interior cells using a cell-centered finite-volume method based on a directional five-cell stencil. W will denote the conservative variables $W = (\rho, \rho u, \rho v, \rho w, \rho E)$ in the following.

The RANS equations are solved with the Spalart–Allmaras turbulence model [43].

2.2 The immersed boundary method

The immersed boundary method described in this paper relies on a ghost-cell direct forcing formulation, deriving from the ideas of Fadlun et al. [19] and Tseng and Ferziger [45]. The flow state values W are imposed at some cell centers located close to the obstacles to mimic a boundary condition taking into account for ghost cells within a body-fitted approach.

These IB or target points are determined at the fringe of solid points, lying inside obstacles. A hole-cutting algorithm, previously used for overset grids [4, 27], is performed to mark interior cells as solid points and exterior cells as fluid points. As displayed in Fig. 1, the Cartesian domain is separated into a solid region below the black curve defining the obstacle and a fluid region above that black curve. As our method relies on a second-order accurate finite-volume solver with a five-point stencil, then the IB target cells are defined by two layers of fringe points surrounding blanked out points. These target cells can lie either inside the obstacle or outside the obstacle. The solution W is reconstructed at these IB target points using information in the fluid close enough to the wall, at *image points*. Figure 2 displays the case where target IB point A (green dot) is inside the obstacle S. For a sake of simplicity, the image point B (in red dot) can be represented as the symmetrical point of target IB point A with respect to the solid boundary. For that purpose, the distance to the obstacles and also the gradient of

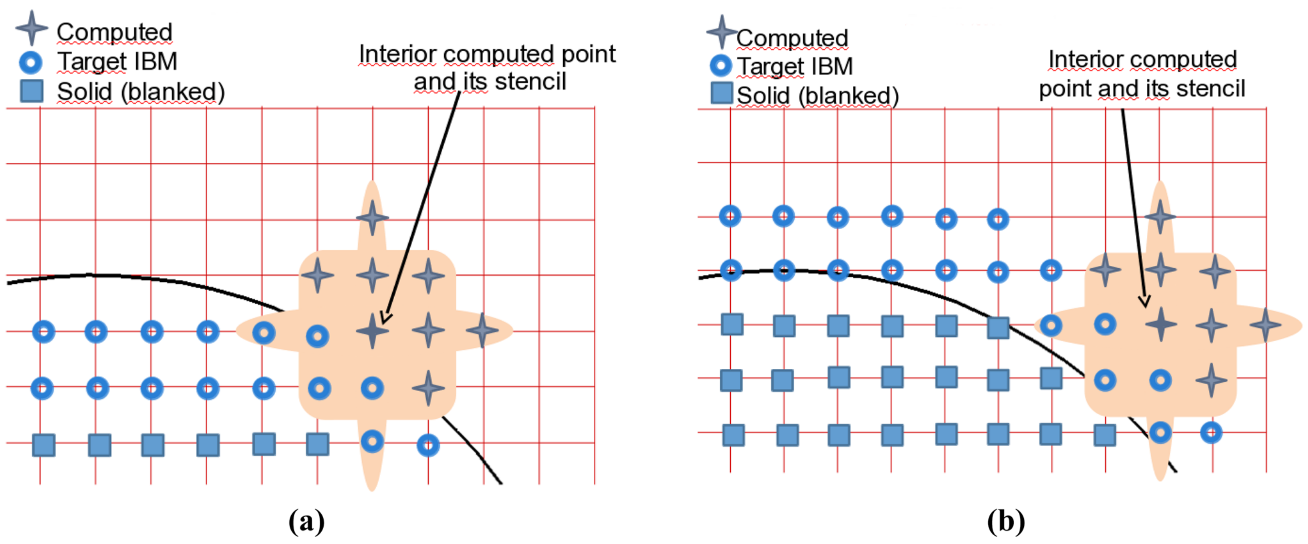

Computed
Target IBM
Solid (blanked)
Interior computed point and its stencil
(a)
Computed
Target IBM
Solid (blanked)
Interior computed point and its stencil
(b)

Fig. 1 Five-point stencil involving IB target points for a viscous simulation: **a** with interior and **b** exterior IB target points

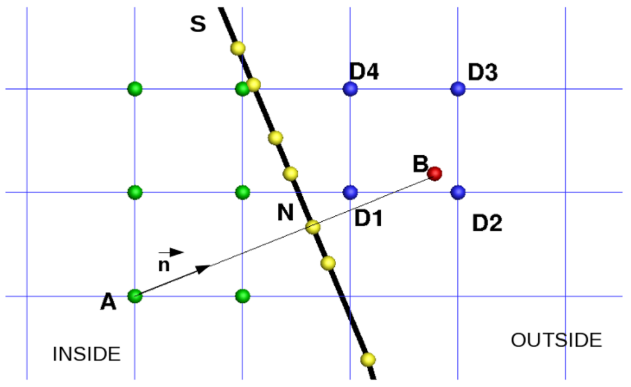


Fig. 2 2D sketch describing the present direct forcing IBM approach. Solution W at IB target point A is built up using the corresponding interpolated value of W at its image point B

the distance to get the normals $\vec{n}$ are required. As depicted in Fig. 2, the image points do not usually match fluid points, and thus, the solution W at point B is obtained by a second-order interpolation using donor points D_1, D_2, D_3, D_4. Point N is the resulting point on the obstacle for which the physical boundary condition shall be recovered implicitly. This point N is obtained by a projection following the normals $\vec{n}$.

2.3 Types of immersed boundary conditions

The reconstruction at IB target points depends on the type of the immersed boundary condition (IBC) defined locally by the input surface.

2.3.1 Wall slip and no-slip IBCs

As displayed in Fig. 2, the image point is not necessary the mirror point of the IB target point with respect to the wall. Thus, a linear reconstruction is applied to recover the boundary conditions at the wall $\vec{u} = 0$ for a no-slip boundary condition and $\vec{u_n} = 0$ for a slip boundary condition.

In the case of a no-slip boundary condition where $\|\vec{u}\| = 0$ at the wall, a one-dimensional linear interpolation is applied given $\vec{u}(B)$ as follows:

$$\vec{u}(A) = \frac{\Delta_{A,N}}{\Delta_{B,N}}\,\vec{u}(B),$$

where $\Delta_{A,N}$ and $\Delta_{B,N}$ are the signed distance of IB target point A and IB image point B to the wall point N, respectively.

In the case of a slip boundary condition, the velocity vector $\vec{u}$ can be decomposed in a normal and a tangential vector as:

$$\vec{u} = \vec{u}_t + \vec{u}_n.$$

The normal velocity vector is obtained by a linear reconstruction, similar to the one applied on $\vec{u}$ for the no-slip boundary condition. The tangential velocity vector is then obtained by $\vec{u}_t(A) = \vec{u}(B) - \|\vec{u}_n(B)\|\vec{n}$, where $\|\vec{u}_n(B)\|$ is the magnitude of the normal velocity at IB image point B and $\vec{n}$ is the normal vector to the wall defined at point A.

Pressure and density gradients are assumed equal to zero in the normal direction to the wall in the close vicinity to the wall; hence: $p(A) = p(B)$ and $\rho(A) = \rho(B)$. The pseudo-viscosity $\tilde{\nu}$ of Spalart–Allmaras one-equation turbulence model is recovered by the same linear interpolation, such that $\tilde{\nu}$ is implicitly zero at the wall.

2.3.2 Wall function for high-Reynolds flow simulations

Our approach relies on an IBM approach on adaptive Cartesian grids, leading to prohibitive cell counts to resolve the viscous stress in the boundary layer until the wall. Moreover, this method is devoted to aeronautical configurations where high Reynolds numbers are often considered. This issue is a key-point addressed by many researchers in the field of IBMs, using a wall function to represent the wall shear stress in the case of viscous flow simulations at high Reynolds numbers [7, 10, 12]. In our approach, Musker's algebraic wall function [31] is used to reconstruct the velocity at IB target points, enabling to place first Cartesian cells near the walls at $y^+ \approx 100$. Details on the wall function are provided in Appendix A. Figure 3 displays the IB target point A and its image point B, for which the variables W_B are interpolated from the computed cells. Instead of a linear reconstruction to recover $u = 0$ at the wall, a wall function is applied between the image point B and the wall. Similarly to the slip boundary condition, the velocity vector is decomposed into a tangential and a normal vector. The tangential velocity vector at point A is obtained as follows:

$$\vec{u}_t(A) = \frac{U_A}{U_B}\vec{u}_t(B),$$

where $U_P = \|\vec{u}_t(P)\|$ denotes the magnitude of the tangential velocity at any point P.

The velocity vector at point B is obtained by interpolation from its neighbouring points. Knowing the modulus of the tangential velocity at image point B, the friction velocity u_τ

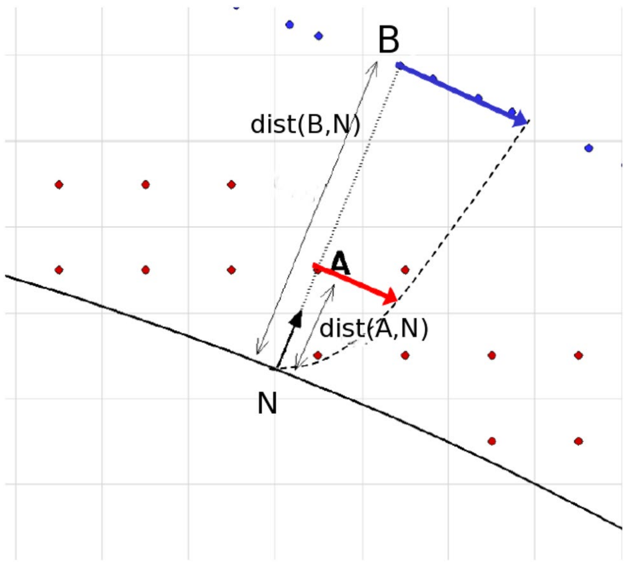


Fig. 3 Wall function for IBM: IB target point is A and corresponding image point is B. In red dots, are IB target points around the obstacle; in blue dots, their image points

is obtained by a Newton–Raphson's method on Musker's algebraic wall function. Then, y^+ at point A is computed by $y^+ = \frac{\Delta_{A,N} u_\tau}{\nu}$. The algebraic function (3) in Appendix A provides the modulus of the tangential velocity vector at point A. The normal velocity at image point B is obtained by a simple projection:

$$\vec{u}_n(B) = (\vec{u}(B) \cdot \vec{n})\vec{n},$$

where $\vec{n}$ denotes the unit normal vector at the wall passing through points A and B.

The tangential velocity at IB image point B can be expressed by:

$$\vec{u}_t(B) = (\vec{u}(B) \cdot \vec{t})\vec{t}.$$

We could have imposed the flow to be locally parallel to the wall, that means $\vec{u}_n = 0$, but this tends in practice to delay the separation on massively separated flows. Thus, a 1D linear interpolation is performed to compute the normal velocity:

$$\vec{u}_n(A) = \frac{\Delta_{A,N}}{\Delta_{B,N}}\vec{u}_n(B).$$

The resulting three components of the velocity vector $\vec{u}$ at point A are then obtained by summing the corresponding normal and tangential vector components.

In the case of an RANS modeling using Spalart–Allmaras model [43], the pseudo-viscosity $\tilde{\nu}$ must also be estimated at IB target point A. Under the assumption of an equilibrium boundary layer, $\tilde{\nu}$ can be defined as:

$$\tilde{\nu} = \frac{1}{f_{v1}}\kappa\, u_\tau y,$$

where f_{v1} is the damping function of Spalart–Allmaras model, which is a nonlinear function of $\tilde{\nu}$, and thus, $\tilde{\nu}$ is also obtained by finding explicitly the root of the resulting quartic equation (it appeared that the Newton's method did not always converge), as detailed in Appendix B.

2.3.3 Inflow and outflow boundary conditions

To perform numerous applications encountered in aeronautics, we have introduced other types of immersed boundary conditions, such as an inflow or an outflow boundary condition; the outflow boundary condition consists in imposing a static pressure field at the corresponding immersed boundary; other values are extrapolated from the values at image point B.

For the inflow boundary condition, the direction of the mean flow $\vec{d}$, the stagnation enthalpy and pressure are defined. The velocity modulus and pressure are obtained by a resolution of non linear equations by a Newton's method. The density and temperature are then derived from all these quantities.

2.3.4 Use of several types of immersed boundary conditions for a given configuration

Several types of immersed boundary conditions can be defined in a single computation, typically to perform a simulation around a model set in a wind tunnel. In that case, a wall function is applied at the boundary of the model, a slip boundary condition at wind tunnel walls, and an inflow and an outflow boundary conditions at inlet and outlet, respectively. The nature of the immersed boundary condition to be applied is determined by the nature of the input surface on which the IB target point A is projected. If the projection point N lies on a surface tagged as an injection immersed boundary, then the injection immersed boundary condition is flagged for IB point A.

2.4 Immersed boundary wall post-processing

Unlike body-fitted approaches, it is not possible to extract the flow fields directly at wall boundaries. A reconstruction must be performed to obtain some quantities such as skin friction or loads and to visualize them on the obstacles. For that purpose, the moving least squares (or MLS) method is performed. It has been initially built up for the generation of surfaces [23] and has been derived to provide spatial approximations for meshless methods [13]. In our approach, the flow quantities (pressure, density, and friction velocity) are extracted at IB wall points at a given iteration of the flow solver. These variables are interpolated using a third-order accurate MLS interpolation onto the vertices of a triangular mesh describing the obstacles on which the skin quantities are required. For each vertex V_T of the tessellation, a point cloud made by at least ten IB wall points surrounding the vertex V_T is used to project the solution on that vertex using the MLS algorithm. Consequently, to introduce interpolation errors due to strong disparities between the distribution of IB wall points and the tessellation (e.g., an MLS stencil ten times larger than the characteristics length of the target triangle), the tessellation should be consistent with the Cartesian mesh discretization in the vicinity of the obstacles. In a recent paper, Capizzano [11] proposed a method to reconstruct the surface to alleviate that constraint and being able to improve the estimation of wall quantities. A simple example of the MLS reconstruction has been achieved for the case of an IBM simulation around a 2D profile. In that case, we can compare the solution at IB wall points directly (represented by red dots in Fig. 4) and the solution obtained after the MLS reconstruction onto the obstacle described by the discretized curve representing the profile.

3 IBM on adaptive Cartesian grids

3.1 Motivation

Most immersed boundary methods available in the literature rely on adaptive Cartesian grids: Cartesian embedded methods remove the bottleneck of the mesh generation, since the adaptive Cartesian mesh generation can be easily automated even for arbitrary complex geometries. To preserve the

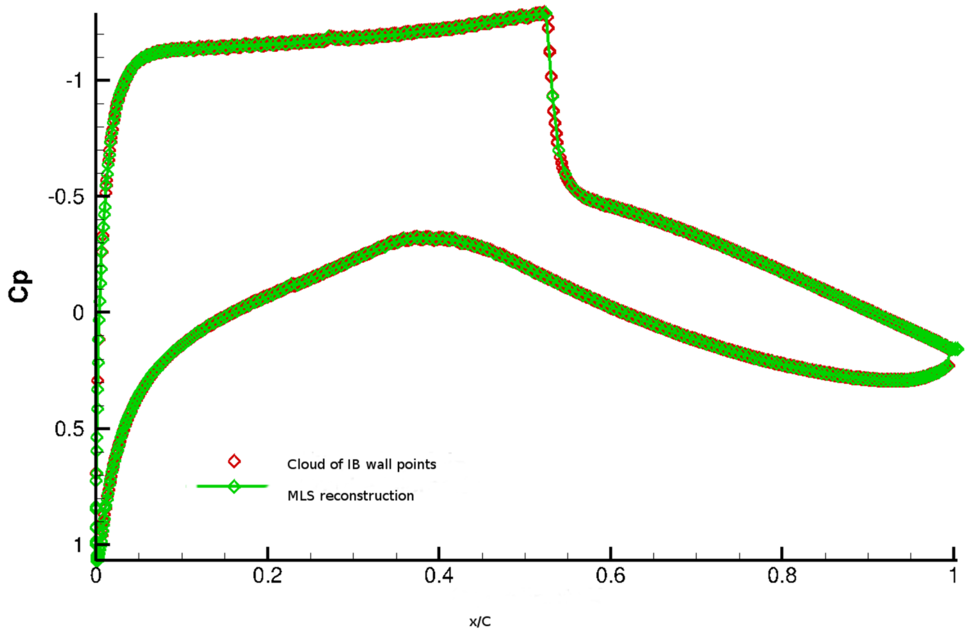


Fig. 4 Comparison of the skin pressure coefficient at IB wall points (red dots) and reconstructed by MLS method on the original discretized profile (green dots)

simplicity of a pure Cartesian approach, the Cartesian mesh is defined down to the wall, relying on the IBM approach to take into account for obstacles. Cartesian cells cannot be refined down to the wall in general (except those cases where the wall is aligned with an axis), so a wall function is mandatory to compute high Reynolds number flows with a reasonable cell count. The strength of the IBM approach on adaptive Cartesian grids used in combination with a Cartesian CFD solver provides an automated and efficient tool for the simulation of flows around complex geometries, provided that the IBM pre-processing is robust and fast.

3.2 Automatic IBM preprocessing for complex geometries

3.2.1 Description of the workflow

The IBM preprocessing can be separated into the following steps:

- The automatic Cartesian mesh generation from a discretized CAD.
- The computation of information required for the IBC reconstruction at each time step of the flow simulation.

First, a Cartesian mesh is generated automatically. This mesh is made of a set of structured uniform grids. The different refinement levels are managed thanks to an octree structure [33], enabling to prescribe the mesh resolution near each boundary and within the fluid, to avoid coarsening in the wake for instance. Ghost cells are explicitly built, such that an overlapping exists between neighbouring grids, with a minimum overlap. An example of a Cartesian mesh generated around a 2D profile is displayed in Fig. 5. To generate that case, the input data are a 1D discretization of the profile and the cell size required in its vicinity (equal to 0.1% of the chord length here). The Cartesian mesh skeleton is a quadtree mesh, as displayed in Fig. 5a. Each element of the quadtree is then filled with a Cartesian grid of a constant number of cells per direction (specified by the user), resulting in an adaptive Cartesian mesh displayed in Fig. 5b. As shown on this figure, some grids that are entirely inside the solid are removed, to reduce memory requirements. The IBM preprocessing is then achieved, based on several geometrical algorithms initially developed for overset grids [4]. Some of the steps are illustrated by the IBM preprocessing of the previous NACA0012 configuration.

- Interior cells are marked using a blanking technique, either using the X-ray technique introduced by Meakin [27] or by a line-of-sight algorithm [4]. Figure 6 displays the same Cartesian mesh, as displayed in Fig. 5b, but where blanked points are not represented in the figure.
- the signed distance field is then computed;
- IB target points are marked at the fringe of blanked points (green dots in Fig. 7a);

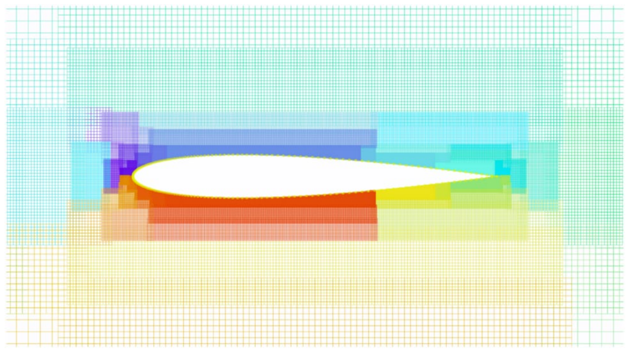


Fig. 6 IBM preprocessing of an NACA0012 profile: blanking of cells inside the obstacle

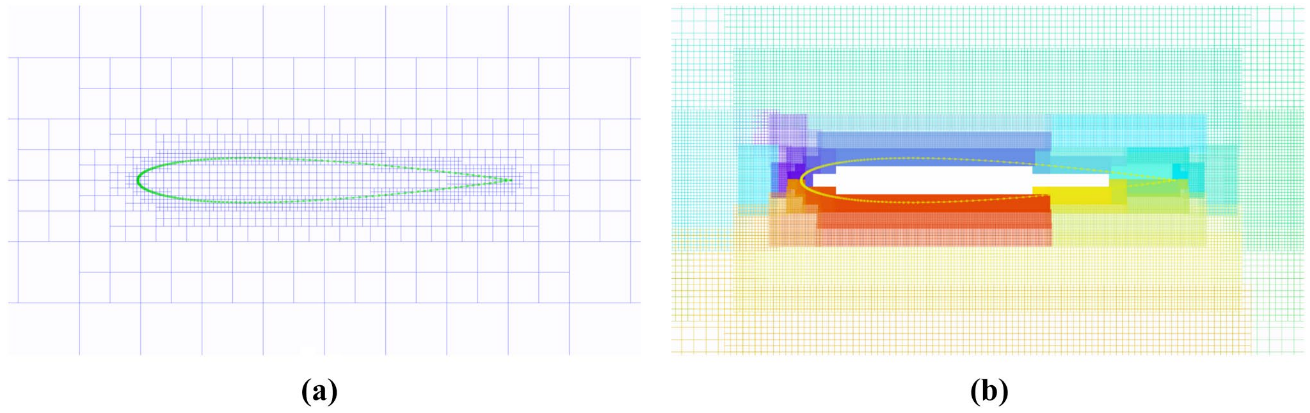


Fig. 5 Example of a mesh around an NACA0012 profile: **a** quadtree skeleton mesh in blue, profile in green; **b** resulting Cartesian mesh, made by 195,000 cells on 104 grids

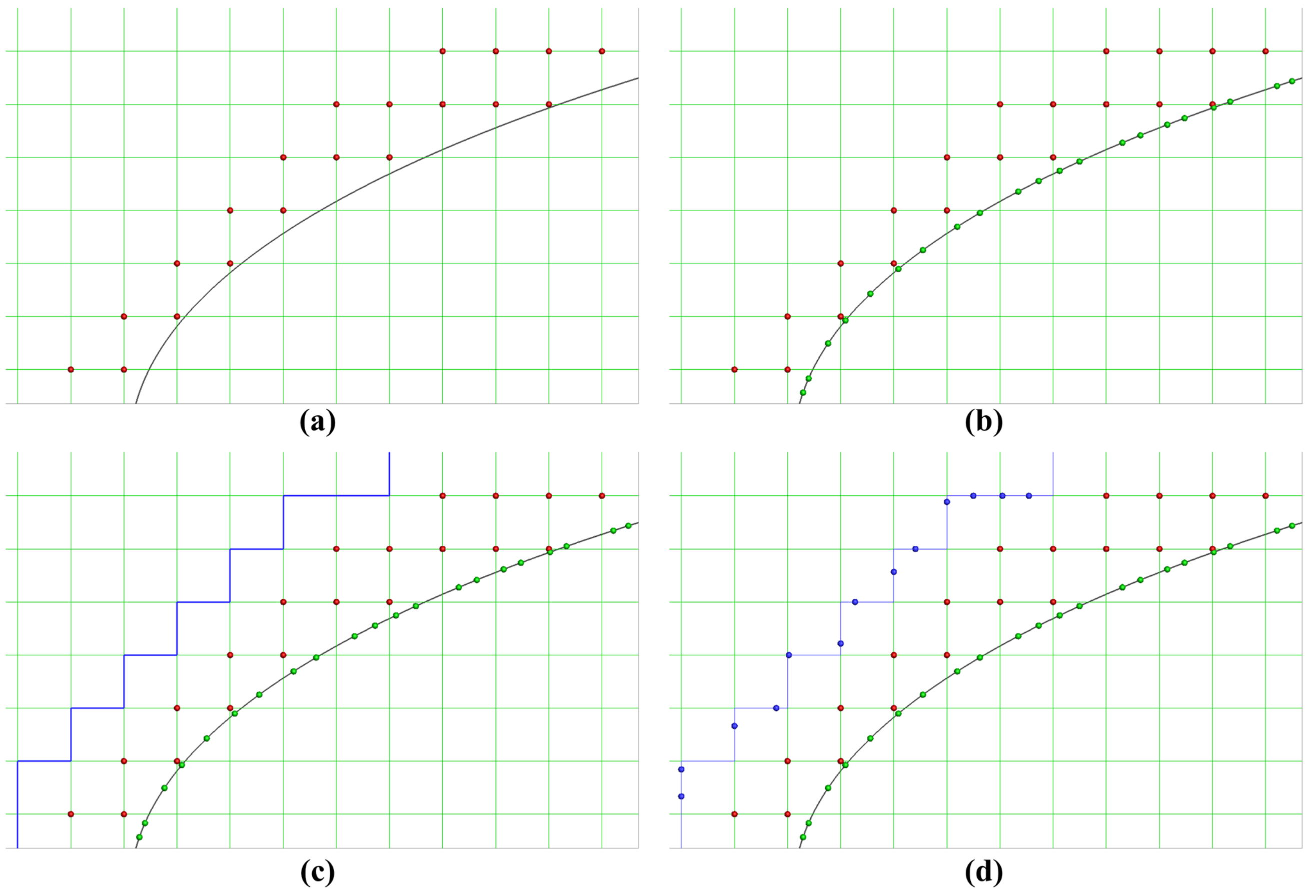


Fig. 7 Closeup view near the leading edge of an NACA0012 profile: **a** IB target points are represented by red dots and the profile by the grey curve; **b** green dots correspond to the IB wall points, resulting from the projection of IB target points following normals onto the profile; **c** front of first computed cells in blue; **d**blue dots are the IB image points obtained by projection of target points following normals onto the front

- normal vectors at IB target points are computed as the local gradient of the signed distance;
- IB target points are then projected onto the immersed boundaries following the normal vectors, resulting in boundary points (red dots in Fig. 7b);
- the location of image points is determined inside the fluid region (blue dots in Fig. 7d);
- the interpolation data for image points are computed (donor cell indices and weights);

3.2.2 Location of image points

The penultimate step, which consists in determining the location of image points, needs a special care to ensure the robustness of the method. First, the fluid variables are reconstructed at iteration n at IB target points using information at that iteration n at image points, and then, donor cell for the interpolation of the flow variables at image point must contain cells where the solution is already known at iteration n. This means that the donor cell must not contain either IB target points or blanked points. The image points must be outside of a front bounded by the first computed cells at the fringe of IB target points. Moreover, the image points must be close to IB target points, especially for wall-modeled IBM reconstruction, where the image points should be lying in the inner layer of the boundary layer to be consistent with Musker's wall function. For simple convex geometries, such as a cylinder or a 2D profile, the image points can be chosen at a given distance from the wall boundaries. However, for complex configurations, that distance cannot be prescribed easily by an user (due to concavities mostly). As a consequence, in our approach, the image points are obtained by a projection following normals of the IB target points on the front of first valid donor cells, which defines a watertight surface mesh around the obstacles.

3.2.3 Performances

The IBM preprocessing can be used in a parallel environment and takes advantage of the Cartesian topology of grids.

First, donor cell search relies on this topology: the image point of coordinates (x, y, z) can be immediately located within a Cartesian cell, knowing the coordinates (x_0, y_0, z_0) of the starting point of the Cartesian grid, the spacing (h_x, h_y, h_z), and the number of points (ni, nj, nk) in the three directions. Let us denote the IB receptor grid a Cartesian grid containing IB target cells. The grid containing the IB image points is denoted IB donor grid. In some cases, the IB donor grid is not necessarily the same as the IB receptor grid (since some IB image points might not fall on their IB receptor grid) and they may not be on the same processor too. Thus, the data to be sent to the receptor processor (containing coordinates of IBM image points) from the processor containing the candidate IB donor grid are only the coordinates of the first Cartesian mesh point (x_0, y_0, z_0), the spacing (h_x, h_y, h_z), and the three mesh dimensions (ni, nj, nk) describing the candidate donor grids.

In addition, tests to determine whether a grid is a candidate IB donor grid for an IB receptor grid are simple and fast, since the tests are intersections of bounding boxes, which are also simplified for Cartesian grids.

Let us denote NP the number of MPI processes and NT the number of OpenMP threads. The octree skeleton mesh is initially built on all the NP processors starting from the triangular meshes describing the immersed boundaries and the cell spacing required in their vicinity. The mesh is then split into NP parts. Only the ith sub-part of the octree is kept on processor of rank i. Since the octree mesh is the skeleton of block-structured Cartesian grids, less memory is required to generate that mesh: to generate a 1 billion point Cartesian mesh with 20 cells per direction in each elementary Cartesian grid, the octree mesh does not exceed 125,000 elements. This has the advantage to balance the octree with a minimum effort.

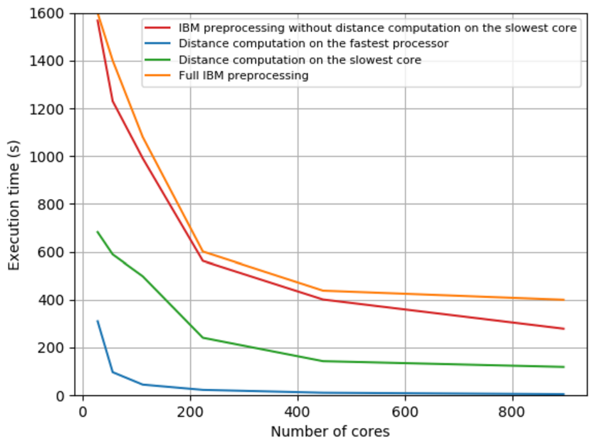
The subset of the octree mesh on a given processor rank i is used to generate local Cartesian grids: in our approach, introduced in [33], the HEXA mesh elements are subdivided into $v_{\min}$ cells, each element resulting in an elementary structured Cartesian grid. Elementary grids are then merged using Rigby's algorithm [39], adapted for Cartesian grids. The cell count on all the processors is roughly the same, ensuring a good balancing between the different processors.

Since surface meshes describing the obstacles are loaded on all the processors, then the blanking and distance field computation can be achieved independently; the identification of target points are also local to the processors. The front of first computed cells is built on Cartesian grids defined locally, and hence, a gathering of the front is performed on all the processors, such that the IB image points can be projected on the front safely.

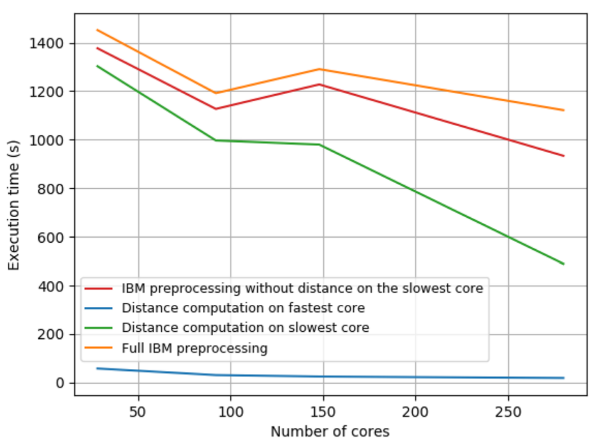
Interpolation data for the IB image points are computed locally on the receptor processor, as described above.

An evaluation of the performances of the IBM preprocessing has been performed. The chosen geometry is a simplified landing gear [25], where the diameter of the wheels is dimensioned to $D = 1$. The triangular mesh defining the landing gear surface is made by 141,696 triangles. Performances are evaluated on Intel Xeon Broadwell partition of the Sator cluster of ONERA (E5-2680v4, 2.4 GHz, 35 MB cache), with 28 cores of 4 GB per node.

A strong scaling evaluation for the IBM preprocessing is achieved on a mesh made of 717 million points, where the number of cores varies from 28 to 896 and the problem requires at least 117 GB for the 28-core test. Figure 8a represents the elapsed time versus the number of cores for that case. The orange curve describes the elapsed time for the whole IBM preprocessing (including the generation of the 717 million point mesh).



(a) Strong scaling study: generated mesh of 717 million points



(b) Weak scaling study: 5 million points per core

Fig. 8 Performance study of the IBM preprocessing for a landing gear configuration

As the number of cores is doubled, the elapsed time is reduced by 12% from 28 to 56 cores, 23% from 56 to 112, 44% from 112 to 224 cores, 27% from 224 to 448 cores, and, finally, 8% from 448 to 896 cores. The 112-core case (which is the most likely value to perform then the CFD computation) requires 16 min to generate the mesh and to perform the IBM preprocessing.

On this plot are also represented the required elapsed times to compute the distance for the slowest and fastest processors (blue and green lines, respectively), showing strong discrepancies between them. The same observation can be done for the weak-scaling study (Fig. 8b). However, this plot highlights the fact that the elapsed time for the full preprocessing is roughly 20 min whatever the size of the whole problem provided 5 million points are defined per core.

It can be noticed that for this test case, the wall-distance computation represents a significant part of the elapsed time of the whole IBM preprocessing (from 90% for the smallest problem to 43% for the biggest problem). The current algorithm consists of an orthogonal projection onto the tessellation defining the immersed boundaries. The search for the candidate triangle is preconditioned by a k–d tree [5] and a boxtree [3], but these preconditioning trees are defined on each processor for the whole tessellation. Consequently, the distance field computation relies strongly on the cell count on the surface meshes despite the number of Cartesian grid points is constant on a processor as the number of processors increases. In addition, the slowest processors for the distance computation are those containing Cartesian grids that are the furthest from the immersed boundary, due to the fact that the number of candidate triangles for the projection is much higher for further points. As the processors that are the slowest regarding the distance are not involved in computing the location of IB points and interpolation data for image points, the orange curve is not the sum of the red and green curves.

Future work will consist in partitioning the surfaces onto the processors and, thus, the corresponding preconditioning trees. Another idea is to solve an Eikonal equation, using, for instance, the *Fast Iterative Method* [22] or the *Fast Marching Method* [42].

IBM preprocessing is achieved by an assembly of Cassiopee functions available in several modules (see reference [4] for a general description of Cassiopee or the website [1]).

3.3 IBM simulations using a dedicated Cartesian CFD solver

3.3.1 FastS HPC solver

The ONERA HPC FastS solver [2] is used to solve the compressible Navier–Stokes equations using a finite-volume method. It contains a structured multiblock solver that can solve RANS, LES, DNS, and steady and unsteady simulations. It is especially efficient to deal with unsteady simulations (see [16]), since it enables to update 10 million cells per second per core on a single Intel Broadwell core. This means that 300 million cells can be updated per second on a 28-core node. FastS contains a solver dedicated to Cartesian grids, on which we rely on to perform IBM simulations. Despite the relatively high cell count obtained by the block-structured Cartesian mesh generation in comparison with a classical body fitted unstructured approach, a dedicated Cartesian solver requires much less memory and CPU time than a structured curvilinear solver and also an unstructured solver. Here, the Cartesian solver is 2.5 more efficient in terms of CPU time and memory than the structured curvilinear solver using the same numerical methods.

FastS solver relies on an hybrid MPI/OpenMP framework, where the memory is distributed (by distributing CFD grids) on the processors at high level, i.e., between nodes, whereas multithreading is managed via OpenMP within a given node. For our purpose, where Cartesian grids are uniform and containing a few cells in comparison with grids resolving boundary layers accurately, the N Cartesian grids are distributed between the NT cores using OpenMP.

3.3.2 Numerical methods

For RANS computations, two spatial schemes are considered, depending on the flow regime: the Roe-MUSCL scheme [40] or an AUSM scheme [26], which is based on a modification of the AUSM+(P) scheme (see Edwards and Liou [18]), which is second-order accurate. Jacobian approximations are those proposed by Jameson and Yoon [21] and Coakley [14], whereas the linear system is solved by the LU-SGS method [21]. For LES computations, an hybrid centered/upwind scheme [26] is retained to manage a good compromise between robustness and accurate simulation of the turbulent small eddies [24], whereas the temporal integration is achieved by a three-step Runge–Kutta explicit scheme, or by a second-order implicit Gear scheme with local Newton sub-iterations [17].

The steady and unsteady RANS equations are solved using Spalart–Allmaras one-equation turbulence model [43]. For large Eddy simulations (LES), the filtered equations are obtained using the formalism developed by Vreman [46]. No subgrid-scale model is used, so it is an implicit LES simulation (ILES).

3.3.3 Update of IBM points during the CFD simulation

The IBM target points must be updated at each sub-step of the time integration. FastS solver updates first fluid cells on each Cartesian grid at time sub-step n, and then, IB target cells are updated, and finally, transfers between neighbouring grids are performed to update the ghost cells. For RANS and LES IBM simulations, Musker's wall model is applied at IB target cells only.

MPI transfers between nodes are achieved in a single step: a global transfer to update all the target points and the ghost cells. This is possible, because the IBM pre-processing prevents from IB image points to be interpolated by ghost cells (which are explicitly defined in the Cartesian mesh).

In practice, only fluid points are computed by FastS CFD solver, transfers between abutting grids and IBM updates are performed by a library of Connector module of Cassiopee package [4]. Both FastS and Cassiopee modules handle the same CGNS/Python tree in memory [36, 41]; in other words, arrays defining the CFD simulation (metrics, flow fields) are shared in memory without copy. This is made possible by the fact that ghost cells are explicitly built during the mesh generation, justifying the use of an overset Cartesian mesh, with minimum overlapping.

4 Numerical results

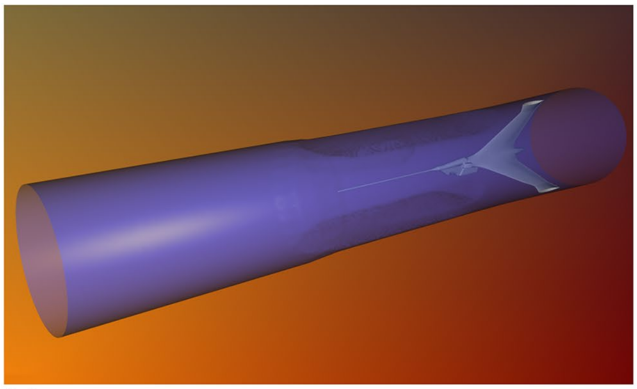
A wide range of validations and applications can be found in [38], showing the range of possibilities, from Euler to LES simulations, from subsonic to hypersonic flows on two-dimensional academic configurations and onto geometrically complex configurations. Here, we focus on two applications: the first one is a tripod mount into ONERA S1MA wind-tunnel, to assess the capability of our IBM approach to perform RANS simulations on a complex configuration within a day. The other test case that has been chosen is an unsteady simulation of a less complex geometry, but where the flow features are complex, to enhance the HPC capabilities of the whole workflow and especially of the flow solver. Despite several imperfections of the present IBM approach on Cartesian grids to capture accurately the physics, especially the acoustics, these first results are promising.

4.1 RANS simulation of the S1MA wind tunnel with a tripod

The test case considered here is a simulation of the ONERA S1MA wind tunnel with a geometrically complex mounted system. This configuration has already been studied numerically and compared with experimental data using a structured body-fitted approach by Hantrais-Gervois et al. [20], to assess the capability of RANS simulations to model the flow physics of an empty wind tunnel with a closed test section. Here, our objective is to demonstrate the capability of the IBM approach on Cartesian adaptive grids to obtain accurate results at low computational cost. In addition, this test case is geometrically complex and makes use of different immersed boundary conditions (injection, outlet pressure condition, and wall-modeled immersed boundary).

The generated mesh is coarse and is made of 35 million Cartesian cells, as displayed in Fig. 9. A steady RANS simulation using Spalart–Allmaras turbulence model is performed. Musker's wall function is applied at IB target points to reconstruct the velocity and wall stress. An injection and imposed pressure immersed boundary conditions are applied at inlet and outlet borders. For that purpose, the triangulated surface defining the wind tunnel is closed; Cartesian grids that lie outside of this closed surface are blanked out.

The target Mach number is 0.8; the outlet pressure is modified to reach a Mach number of 0.79. A comparison between the solution obtained with the IBM on Cartesian grids and the structured body-fitted simulation with elsA software [9] is achieved described in detail in [20]. Mach number contours compare well on two axial sections (at y=0 and z=0, Fig. 10), the slight differences being due to the fact that the actual Mach number is slightly different. Figure 11 displays the pressure evolution along the axis of the wind tunnel and on the tripod, showing a good agreement between the IBM simulation and the reference


(a) Immersed boundaries: wind tunnel wall, inlet and outlet boundaries and tripod surface.

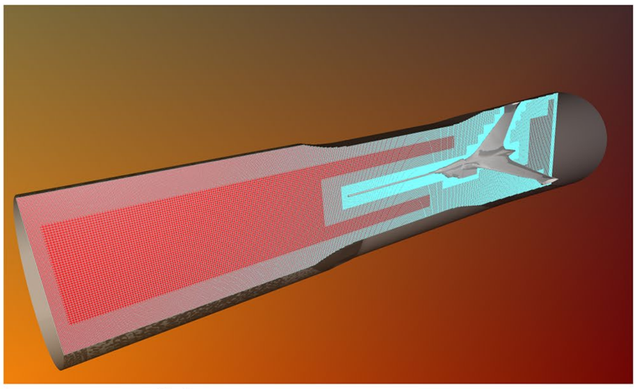
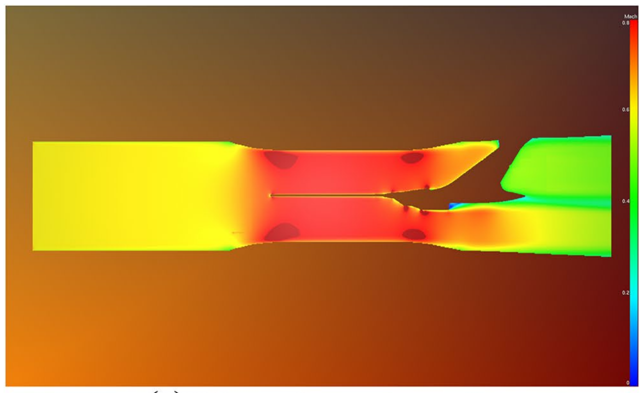
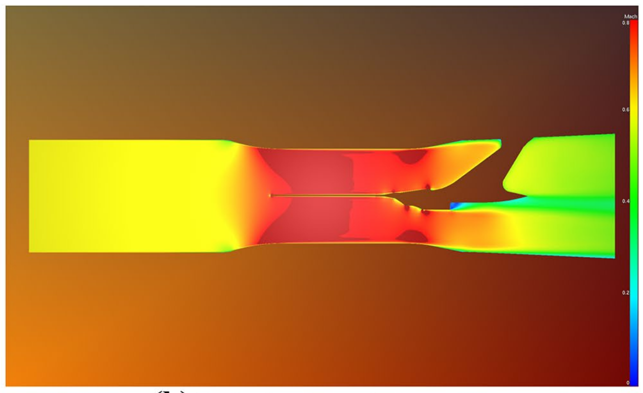

(b) Resulting Cartesian mesh.

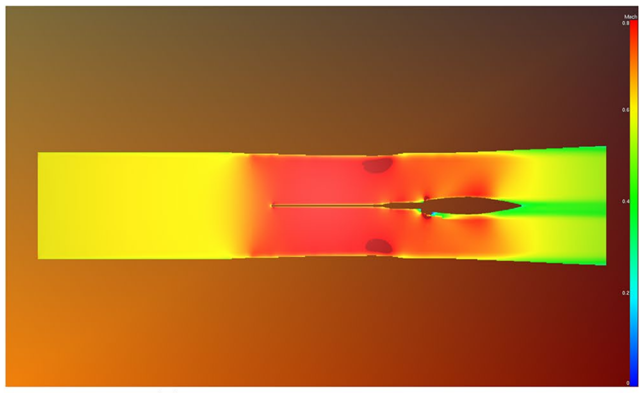
Fig. 9 S1MA wind tunnel with a tripod



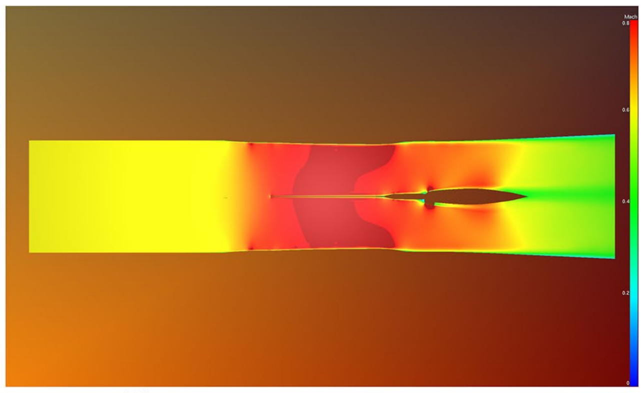
(a) IBM simulation: plane y=0.



(b) Body-fitted simulation y=0.

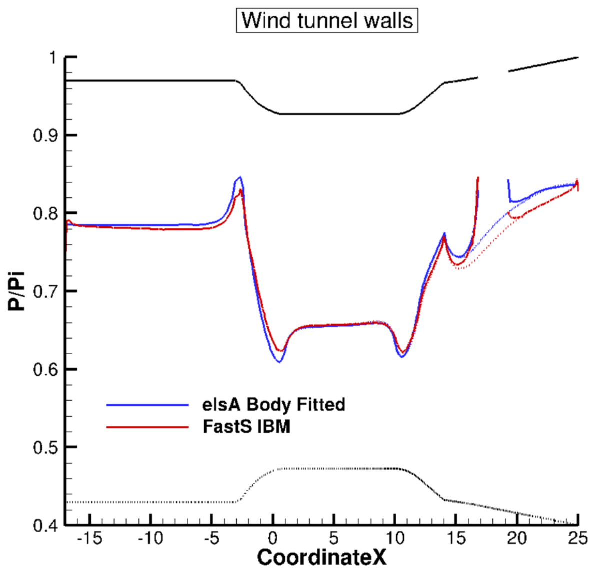


(c) IBM simulation: plane z=0.

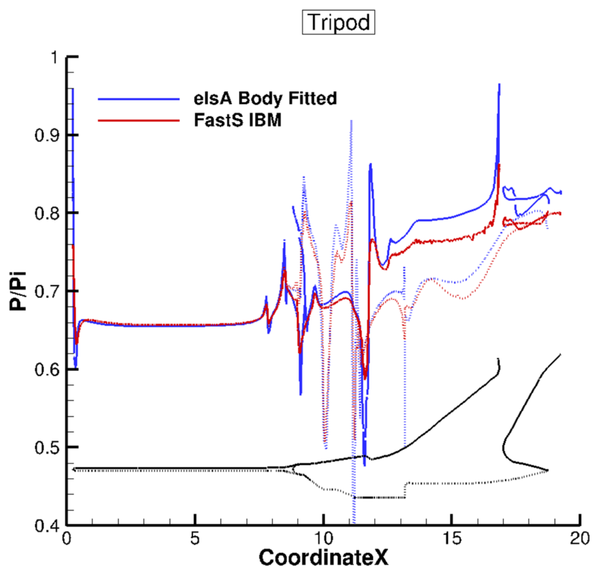


(d) Body-fitted simulation: plane z=0.

Fig. 10 Comparison of Mach number contours (Cartesian IBM solution versus structured body-fitted solution



(a) on wind tunnel boundaries



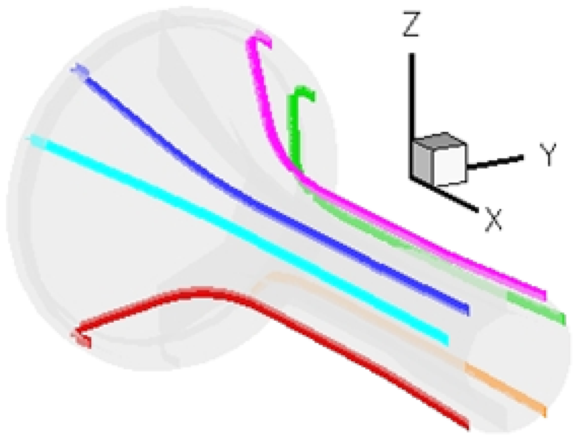
(b) on the tripod surface

Fig. 11 Comparison of the pressure along the axial direction on the wind tunnel; dashed lines denote the lower side of the tripod or of the wind tunnel. Courtesy of Aurélia Cartieri, ONERA/DS for *elsA* results

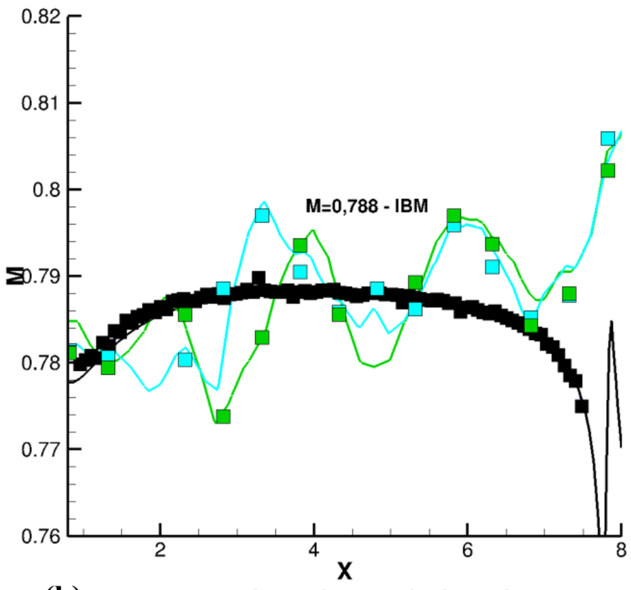
body-fitted simulation. However, some discrepancies can be observed at the junction between the upper side of the tripod and the top of the wind tunnel (Fig. 11b): they are probably due to the fact that the IBM Cartesian mesh is too coarse in that region. Consequently, as the boundary forms a 45° angle with the Cartesian axis, the fact that the mesh is not well resolved in that region leads to a stair-step reconstruction at tripod wall boundaries. Figure 12 displays a comparison of the isentropic Mach number on the wind tunnel walls between experimental results and IBM results. The oscillations observed in experiments, which are due to the waviness of the walls, as explained by Hantrais-Gervois et al. [20], are well captured by the IBM simulation.

4.2 Unsteady-flow simulation around a high-lift airfoil

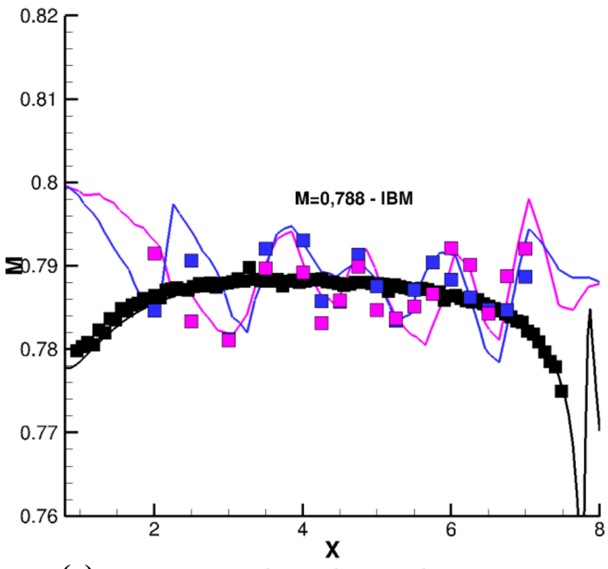
The test case is an extruded three-element high-lift airfoil with deployed slat and flap. This kind of configuration is of major interest for acoustics, since high-lift devices deployed on aircraft to increase lift at low speed are responsible for a significant part for the airframe noise during the approach phase. An experimental campaign has been conducted in



(a) Location of the extraction curves on the wind tunnel walls



(b) Isentropic Mach number on the lateral curves



(c) Isentropic Mach number on the top curves

Fig. 12 Symbols denote the experimental results and solid lines the IBM results. Wind tunnel walls in colored lines and tunnel centerline in black. Courtesy of Aurélia Cartieri, ONERA/DS

the framework of the joint ONERA/DLR LEISA2 project; experimental data are also provided within the AIAA BANC workshops to validate the numerical methods applied for aerodynamics and acoustics analyses. A reference study is the LES simulation of Terracol and Manoha [44] on a 2.6 billion body-fitted mesh. Six million hours of CPU were required on 4096 processors to perform this simulation. This simulation has also been performed by LBM solvers using an IBM approach on Cartesian grids.

The aim of the simulation presented here is to focus only on the aerodynamics phenomena and not on the acoustics, since the way which the information is transferred from a grid of a level *l* to a grid of a different level (twice as coarse) leads to small perturbations that are a major issue for a far-field acoustics analysis.

4.2.1 Description of the test-case

The reduced geometry configuration is used here (F16). The retracted wing chord length is 300 mm. The slat and flap are deployed, respectively, of 27.834° and 35, 011°. The flow conditions are $M_\infty = 0.178$, $\alpha = 6.15°$ and a Reynolds number of Re = 1.23 million, based on the chord. The wing span is chosen the same as the reference CFD study, that is 0.25 *c*.

An IBM simulation with FastS solver is performed on a set of Cartesian grids using Musker's wall function applied at IB target points. The mesh is composed by 660 million points, with an adapted spatial resolution in the vicinity of the flap and the slat and in their cavity and wake regions, with a smallest cell size equal to 1.5 10^{-4} *c*. The external border of the computational domain is located at 50 *c*. The mesh is represented on different views displayed in Fig. 13.

The LES simulation has been initialized by an RANS simulation to get rid of transient phenomena. The spatial scheme is the modified AUSM scheme [26], to manage a good compromise between robustness and accurate simulation of the turbulent small eddies [24], whereas the temporal integration is an explicit three-step Runge–Kutta scheme, with a global time step $\Delta t = 0.16$ µs. The simulation has been performed on 224 Intel Broadwell cores of ONERA SATOR cluster, for a CPU cost of 0.4 µs per point per iteration per core. The flow quantities have been averaged on a period of 80 ms.

4.2.2 Results

Figure 14 displays the density gradient resulting from the LES simulation using the wall-modeled IBM. The

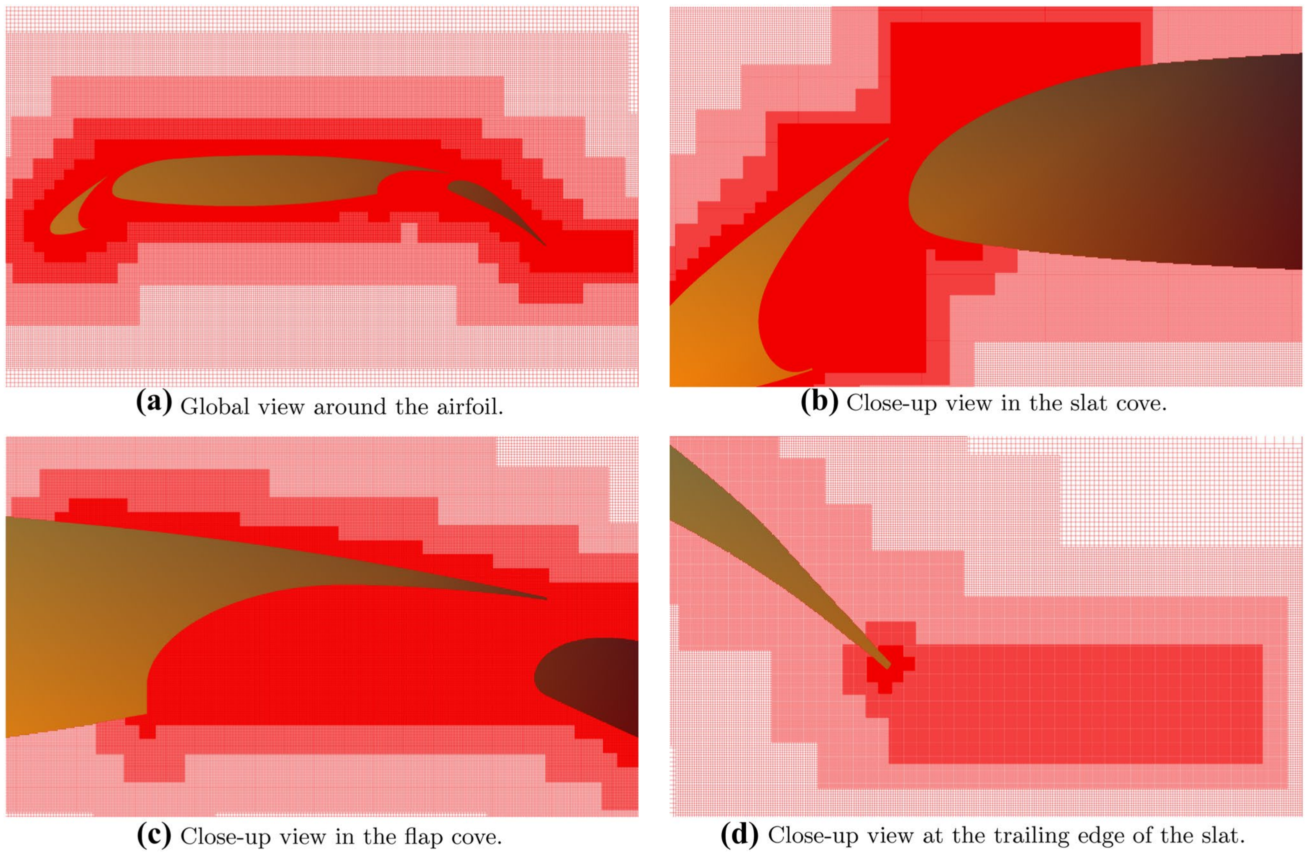
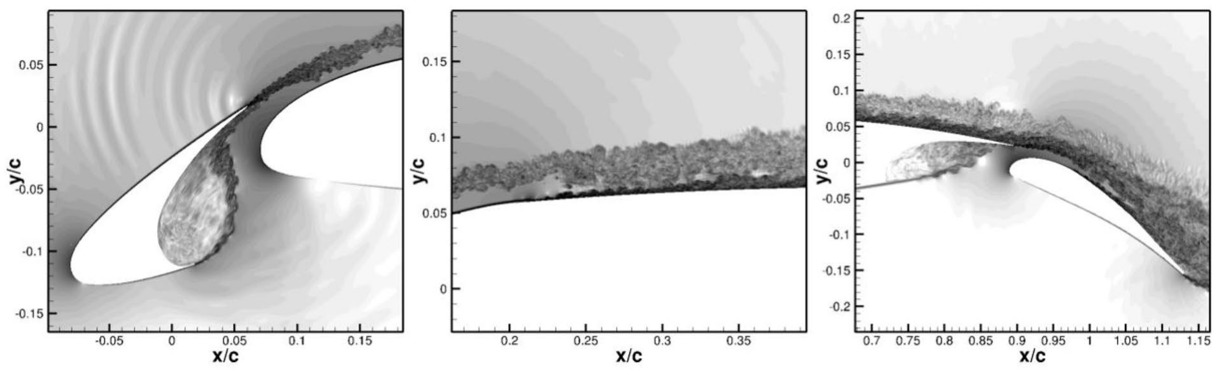
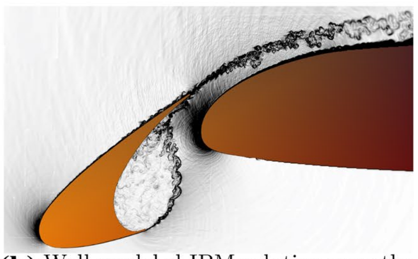

(a) Global view around the airfoil.
(b) Close-up view in the slat cove.
(c) Close-up view in the flap cove.
(d) Close-up view at the trailing edge of the slat.

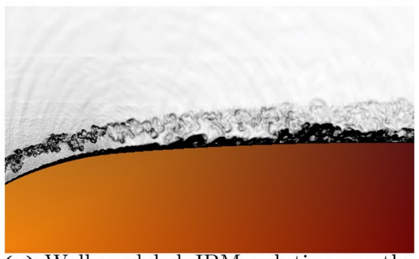
Fig. 13 Views of the IBM Cartesian mesh around the three-element airfoil



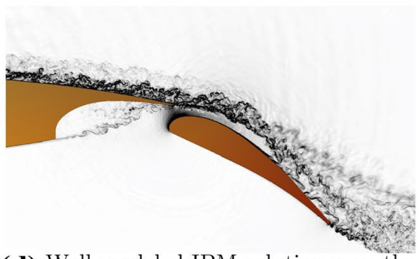
(a) Wall-resolved body-fitted LES solution.



(b) Wall-modeled IBM solution near the slat.



(c) Wall-modeled IBM solution on the main wing.



(d) Wall-modeled IBM solution near the flap.

Fig. 14 Instant views of the flow represented by the density gradient: comparison between the wall-resolved LES (**a**) and the IBM simulations (**b**–**d**)

comparison with the reference simulation of Terracol and Manoha shows that the IBM approach enables to capture the main features of this flow: recirculation bubble in slat and flap cavities, turbulent boundary layers, wakes. This is also assessed by the comparison of the averaged velocity between the reference LES and the IBM simulation and experimental data, as displayed in Fig. 15. The location of recirculation bubble in cavities is well captured. Besides, the simulated flows in the vicinity of the suction side of the flap differ from the experiments, where a strong separation occurs unlike the LES simulations. Other wind tunnel tests did not revealed that separation and Terracol [44] demonstrated that this difference was due to the influence of the wind tunnel walls.

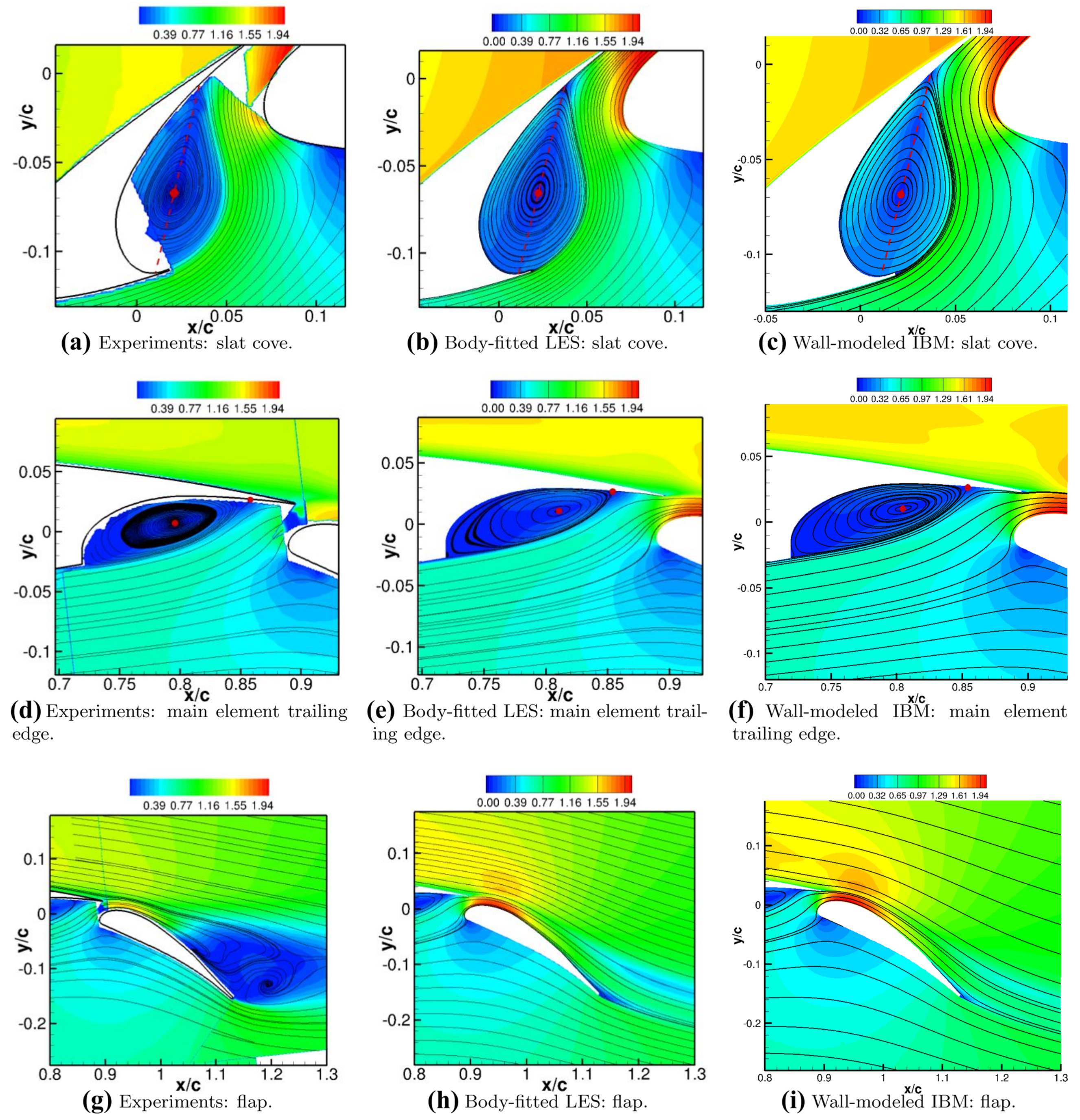
Two rakes of probes are defined in the fluid, as displayed in Fig. 16. At these locations, the velocity and velocity fluctuation profiles are compared against the experiment and the reference LES body-fitted simulation, as displayed in Fig. 17, showing a good agreement between both simulations and also with the experimental results.

5 Conclusions

In this article, we have presented an immersed boundary method (IBM) for compressible flow simulations to evaluate the aerodynamics of complex geometries. Our approach consists of modifying the flow variables at some IB target points in the vicinity of the obstacles. To take a full advantage of this approach, where the mesh does not need to conform to the obstacles, we use adaptive structured Cartesian grids, in combination with a dedicated HPC Cartesian solver, taking advantage of their low memory and CPU time requirements and the automation of the mesh generation and adaptation. This enables us to generate a 1.5 billion node mesh and perform the IBM preprocessing within 18 min on 12×28 cores, requiring a maximum of 360 GB of memory.

The workflow that is built up has demonstrated to be fast, robust, and automated, starting from a discretization of the obstacles only. Consequently, CFD simulations around complex geometries can be performed within a day.

Several types of immersed boundaries have been developed, such that inviscid or viscous wall boundaries can be reconstructed, but also injection and outlet boundaries can be defined as immersed boundaries, provided that the corresponding triangulated surface is defined as input. Turbulent flow simulations are performed with Reynolds-average Navier–Stokes equations using Spalart–Allmaras model or with large-eddy simulation approach, in combination with an algebraic wall function to solve high Reynolds number flows, to mitigate the cell count resulting from the isotropic nature of Cartesian cells.



(a) Experiments: slat cove. **(b)** Body-fitted LES: slat cove. **(c)** Wall-modeled IBM: slat cove.

(d) Experiments: main element trailing edge. **(e)** Body-fitted LES: main element trailing edge. **(f)** Wall-modeled IBM: main element trailing edge.

(g) Experiments: flap. **(h)** Body-fitted LES: flap. **(i)** Wall-modeled IBM: flap.

Fig. 15 Views of the averaged flow: isocontours of the velocity amplitude and streamlines; comparison between experiments (left-hand side), the reference wall-resolved LES simulation (center), and the IBM LES simulation (right-hand side)

The first application shown (the tripod mounted in ONERA-S1MA wind tunnel) demonstrates the robustness and automation of the approach developed here to perform an RANS simulation of a complex configuration, involving several types of immersed boundaries. Results compare well with a structured body-fitted approach achieved with elsA solver [9]; this demonstrates that the method is a good candidate to calibrate quickly wind tunnel configurations, with no meshing effort.

The second application is an unsteady simulation of the flow around a high-lift airfoil. Only aerodynamics results are evaluated here and compared with experiments and a reference LES simulation on a structured body-fitted mesh by Terracol [44]. Acoustics analysis is not performed here,

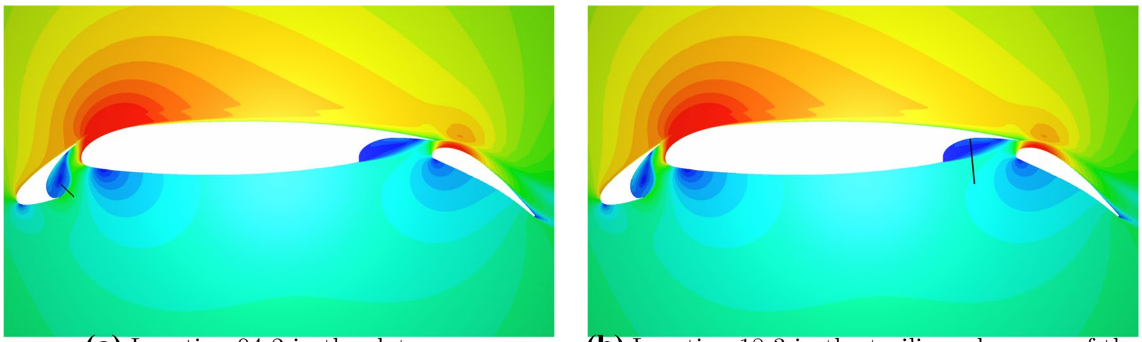


Fig. 16 Probe locations

since no specific treatment is achieved yet when crossing an interface from a fine grid to a coarser grid (twice as coarse here), leading to reflections of unsupported structures back into the finer grid. This is one subject on which we will focus on in the next years.

Future work will also concern the improvement of the wall modeling using wall functions, since the wall function that we consider here can be applied on attached flows.

Another topic is to extend the method to bodies in relative motion, aiming at simulating flows around configurations with rotors.

Acknowledgements We thank Aurélia Cartieri from the Wind Tunnel Division at ONERA for providing us the mounted tripod configuration and her *elsA* results for comparisons. We are grateful to Nicolas Alferez for his involvement in improving the performances of FastS solver and also Marc Terracol for our fruitful exchanges concerning wall models and the LEISA2 configuration.

Wall functions

Figure 18 shows a typical mean velocity profile in wall units u^+ within the inner layer of a turbulent boundary layer. This velocity profile can be split into three portions within this inner layer:

- The viscous sub-layer, for $y^+ \leq 5$, where dissipation and viscous diffusion dominate. This yields the linear behavior: $u^+ = y^+$.
- the log-layer, for $y^+ > 30$, where there is an equilibrium between turbulence production and dissipation. This region constitutes the junction between the inner and upper layers.
- The buffer layer, for $5 < y^+ \leq 30$, joining the two previously defined layers.

The most common function to describe the evolution of the velocity within an equilibrium turbulent boundary layer (zero-pressure gradient) is the log law of the wall defined as:

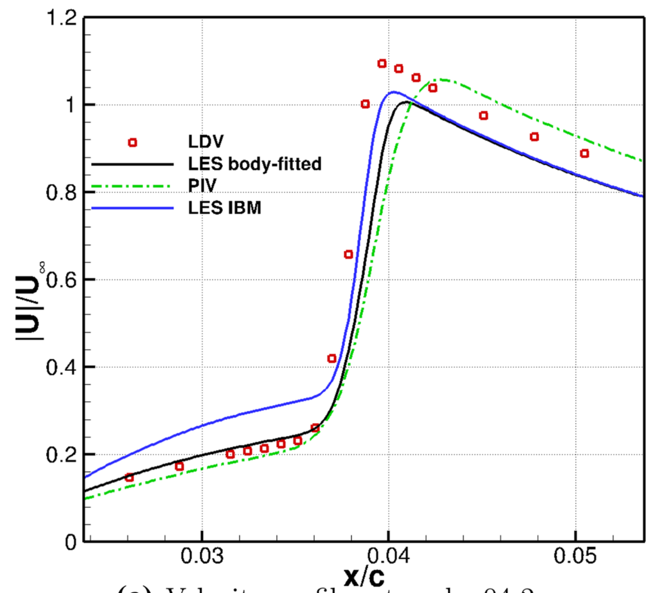
$$u^+ = \frac{1}{\kappa}\log(y^+) + \beta, \tag{2}$$

where $u^+ = \dfrac{u}{u_\tau}$ and $y^+ = \dfrac{\rho_w\, y\, u_\tau}{\mu_w}$, with $\kappa = 0.41$ is the Vón Kármán constant and $\beta = 5.2$; u_τ denotes the friction velocity; ρ_w and μ_w denote the values of density and viscosity at the wall, assumed equal to their values at corresponding image points B.

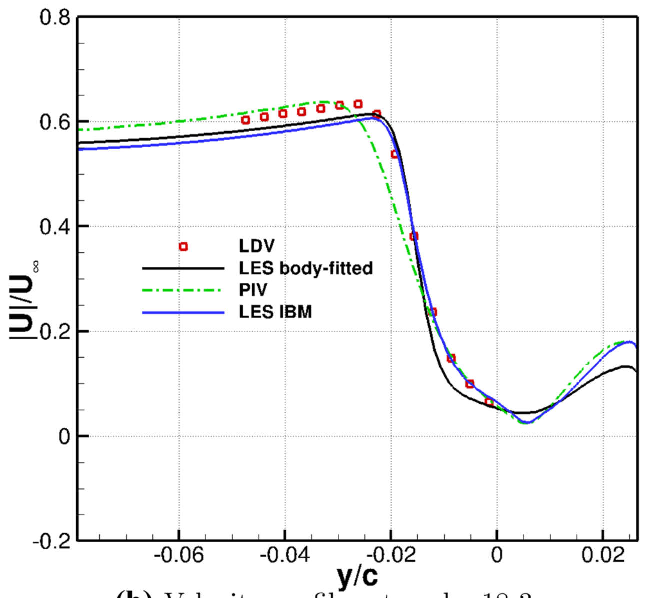
However, the limitation of the log law is that it is not able to model the inner and buffer layers of the boundary layer, which is critical in our approach, since the dimensionless wall distance y^+ cannot be controlled at image points. Several algebraic wall functions have been developed to bridge the viscous sub-layer and the log layer: we can cite the law derived by Spalding, by finding a power-series for $y^+ = f(u^+)$ or the one proposed by Musker [31], which is very similar (as shown in Fig. 18) but easier to use, since it explicitly provides an expression for the velocity for the point to be addressed. Similarly to the log law, it is based on considerations of the boundary-layer equations. By blending the log layer and the viscous sub-layer asymptotic trends of the turbulent viscosity through an interpolation function, integration of the momentum balance yields the following formula:

$$u^+ = 5.424\arctan\left[\frac{2y^+ - 8.15}{16.7}\right] + \log\left[\frac{\left(y^+ + 10.6\right)^{9.6}}{\left(y^{+2} - 8.15\,y^+ + 86\right)^2}\right] - 3.52. \tag{3}$$

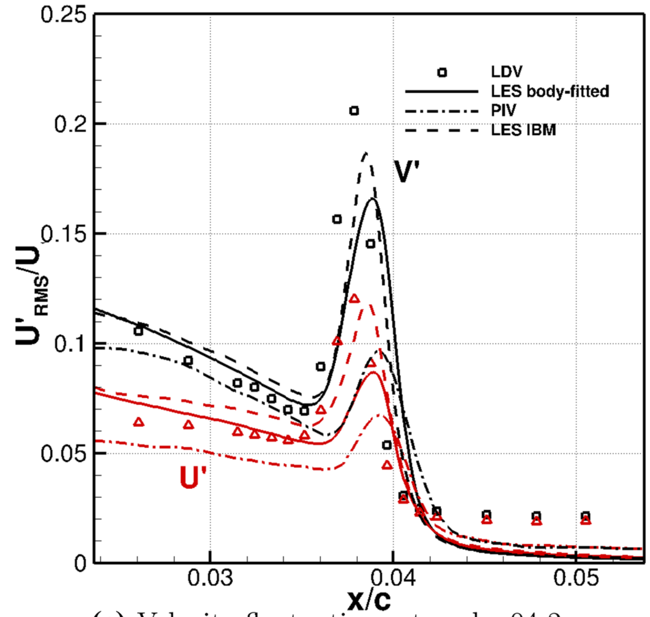
It must be noted that expressions (2) and (3) involve the skin friction velocity u_τ, which is unknown. The first step of the process is, therefore, to estimate its value using a Newton–Raphson iterative algorithm.



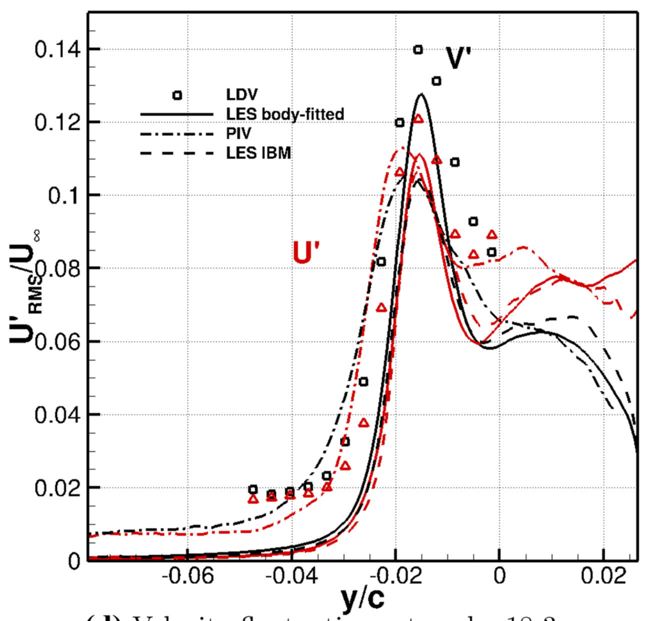
(a) Velocity profiles at probe 04-2.



(b) Velocity profiles at probe 18-3



(c) Velocity fluctuations at probe 04-2.



(d) Velocity fluctuations at probe 18-3.

Fig. 17 Comparison of velocity profiles and velocity fluctuations at probes 04-2 and 18-3. IBM simulation is compared against the reference LES simulation, PIV, and LDV data

Determination of the pseudo-viscosity of Spalart–Allmaras at IB target point

In the Spalart–Allmaras model, the turbulent viscosity can be expressed as follows:

$$\mu_t = \rho \tilde{\nu} f_{v1}, \tag{4}$$

where:

$$f_{v1} = \frac{\chi^3}{\chi^3 + C_{v1}^3}; \tag{5}$$

C_{v1} is a constant and $\chi = \frac{\rho \tilde{\nu}}{\mu}$. Hence:

$$\nu_t = \tilde{\nu} f_{v1}. \tag{6}$$

The mixing length assumption can be expressed by:

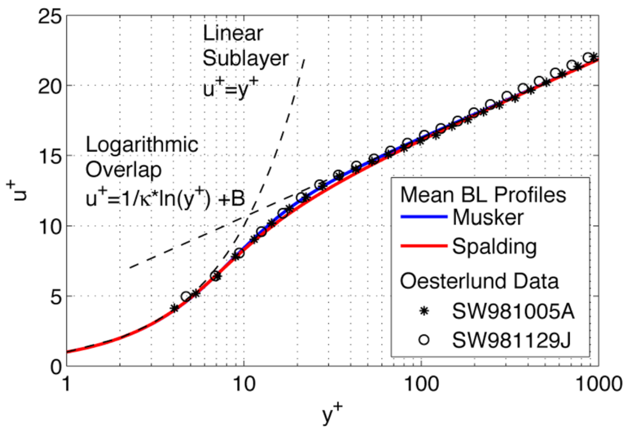


Fig. 18 Asymptotic behaviors of an equilibrium boundary layer

$$\nu_t = \kappa u_\tau y D, \tag{7}$$

with the Van Driest damping term D, such that A^+ being a constant, chosen equal to 19:

$$D = [1 - exp(-\frac{y^+}{A^+})]^2. \tag{8}$$

The pseudo-viscosity $\tilde{\nu}$ must be reconstructed at IB target point A. The friction velocity u_τ is known and has been computed by the algebraic wall function, and y and D are known and κ is the Von Kármán constant, equal to 0.4. We have to solve $\tilde{\nu}$ solution of:

$$\tilde{\nu} f_{v1} = \kappa u_\tau y D; \tag{9}$$

that is:

$$\tilde{\nu}^4 - \kappa u_\tau y D \tilde{\nu}^3 - \kappa u_\tau y D \frac{\mu^3}{\rho^3} C_{v1}^3 = 0. \tag{10}$$

To avoid ill-conditioned problems, the variable that is actually solved is $\frac{\tilde{\nu}}{\nu}$. This leads to solve:

$$x^4 - ax^3 - b = 0, \tag{11}$$

with:

$$x = \frac{\tilde{\nu}}{\nu}; a = \frac{\kappa u_\tau y D}{\nu}; b = \frac{\kappa u_\tau y D}{\nu} C_{v1}^3 = a C_{v1}^3. \tag{12}$$

It is possible to solve this equation explicitly. The following variable change $y = x - \frac{a}{4}$ is performed to remove the monomial of degree 3, leading to an equation of the form $y^4 + py^2 + qy + r = 0$, which is solved by Ferrari's method. Note that if Eq. (11) was obtained from variable $x = \tilde{\nu}$, q would be very close to zero. Or if it is zero, the equation to be solved would be of the form $y^4 + py^2 + r = 0$, with different solution from the quartic equation above.

Ferrari's method consists in finding a factorization of two polynomials of degree 2. The main difficulty lies in the fact that four solutions of this equation are possible, and thus, the wrong candidates (especially the complex ones) shall be removed smartly. The monomial of degree 4 is first replaced by the polynomial $(y^2 + \lambda^2)^2 - 2\lambda\, y^2 - \lambda^2$. This leads to the resolution of a cubic on λ, and then, the solution λ_0 is replaced in the quartic on y. This results in a factorization of two polynomials of degree 2. The roots are explicitly obtained, and then, x is derived.

References

1. http://elsa.onera.fr/Cassiopee/Userguide.html. Accessed 5 Feb 2020
2. https://w3.onera.fr/FAST. Accessed 5 Feb 2020
3. Barequet G, Chazelle B, Guibas LJ (1996) BOXTREE: a hierarchical representation for surfaces in 3D. Comput Graph Forum 15
4. Benoit C, Péron S, Landier S (2015) Cassiopee: a CFD pre- and post-processing tool. Aerosp Sci Technol 45:272–283
5. Bentley JL (1975) Multidimensional binary search trees used for associative searching. Commun ACM 18(9):509–517
6. Berger MJ, Aftosmis MJ (2012) Progress towards a Cartesian cut-cell method for viscous compressible flow. AIAA paper 2012-1301
7. Berger MJ, Aftosmis MJ (2017) An ODE-based wall model for turbulent flow simulations. AIAA J 2:1–15
8. Brehm C, Barad MF, Kiris CC (2016) Open rotor computational aeroacoustic analysis with an immersed boundary method. In: 54th AIAA aerospace sciences meeting, p 0815
9. Cambier L, Heib S, Plot S (2013) The ONERA elsA CFD software: input from research and feedback from industry. Mech Ind 14(03):159–174
10. Capizzano F (2011) Turbulent wall model for immersed boundary methods. AIAA J 49(11):2367–2381
11. Capizzano F (2018) Automatic generation of locally refined Cartesian meshes: data management and algorithms. Int J Numer Methods Eng 113(5):789–813
12. Chen ZL, Hickel S, Devesa A, Berland J, Adams NA (2014) Wall modeling for implicit large-eddy simulation and immersed-interface methods. Theoret Comput Fluid Dyn 28(1):1–21
13. Cheng Z-Q, Wang Y-Z, Li Bao, Xu Kai, Dang Gang, Jin S-Y (2008) A survey of methods for moving least squares surfaces. In: Proceedings of the fifth Eurographics/IEEE VGTC conference on point-based graphics, p 9–23
14. Coakley TJ (1985) Implicit upwind methods for the compressible Navier–Stokes equations. AIAA J 23(3):374–380
15. Coirier WJ, Powell KG (1996) Solution-adaptive Cartesian cell approach for viscous and inviscid flows. AIAA J 34(5):938–945
16. Dandois J, Mary I, Brion V (2018) Large-eddy simulation of laminar transonic buffet. J Fluid Mech 850:156–178
17. Daude F, Mary I, Comte P (2014) Self-adaptive Newton-based iteration strategy for the les of turbulent multi-scale flows. Comput Fluids 100:278–290
18. Edwards JR, Liou M-S (1998) Low-diffusion flux-splitting methods for flows at all speeds. AIAA J 36(9):1610–1617
19. Fadlun EA, Verzicco R, Orlandi P, Mohd-Yusof J (2000) Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. J Comput Phys 161(1):35–60

20. Hantrais-Gervois J-L, Cartieri A, Mouton S, Piat J-F (2010) Empty wind tunnel flow field computations. Int J Eng Syst Model Simul 2(1–2):46–57
21. Jameson A, Yoon S (1987) Lower-upper implicit schemes with multiple grids for the euler equations. AIAA J 25(7):929–935
22. Jeong W-K, Whitaker RT (2008) A fast iterative method for Eikonal equations. SIAM J Sci Comput 30(5):2512–2534
23. Lancaster P, Salkauskas K (1981) Surfaces generated by moving least squares methods. Math Comput 37(155):141–158
24. Laurent C, Mary I, Gleize V, Lerat A, Arnal D (2012) DNS database of a transitional separation bubble on a flat plate and application to RANS modeling validation. Comput Fluids 61:21–30
25. Manoha E, Bulté J, Caruelle B (2008) LAGOON: an experimental database for the validation of CFD/CAA methods for landing gear noise prediction. In: 14th AIAA/CEAS aeroacoustics conference, AIAA paper 2008-2816
26. Mary I, Sagaut P (2002) Large Eddy simulation of flow around an airfoil near stall. AIAA J 40(6):1139–1145
27. Meakin RL (2001) Object X-rays for cutting holes in composite overset structured grids. AIAA paper 2001-2537
28. Mittal R, Dong H, Bozkurttas M, Najjar FM, Vargas A, von Loebbecke A (2008) A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. J Comput Phys 227(10):4825–4852
29. Mittal R, Iaccarino G (2005) Immersed boundary methods. Annu Rev Fluid Mech 37:239–261
30. Mochel L, Weiss P-E, Deck S (2014) Zonal immersed boundary conditions: application to a high-Reynolds-number afterbody flow. AIAA J 52(12):2782–2794
31. Musker AJ (1979) Explicit expression for the smooth wall velocity distribution in a turbulent boundary layer. AIAA J 17(6):655–657
32. Nakahashi K (2011) Immersed boundary method for compressible Euler equations in the Building-Cube Method. AIAA paper 2011-3386
33. Péron S, Benoit C (2013) Automatic off-body overset adaptive Cartesian mesh method based on an octree approach. J Comput Phys 232(1):153–173
34. Peskin CS (1972) Flow patterns around heart valves: a numerical method. J Comput Phys 10(2):252–271
35. Peskin CS (2002) The immersed boundary method. Acta Numer 11:479–517
36. Poinot M (2010) Five good reasons to use the hierarchical data format. Comput Sci Eng 12(5):84–90
37. Beyer RP, LeVeque RJ (1992) Analysis of a one-dimensional model for the immersed boundary method. SIAM J Numer Anal 29(2):332–364
38. Renaud T, Benoit C, Péron S, Mary I, Alferez N (2019) Validation of an immersed boundary method for compressible flows. In: AIAA Scitech 2019 Forum, AIAA paper 2019–2179
39. Rigby D L, Steinthorsson E, Coirier WJ (1997) Automatic block merging methodology using the method of weakest descent. AIAA paper 97-0197
40. Roe PL (1981) Approximate Riemann solvers, parameter vectors, and difference schemes. J Comput Phys 43(2):357–372
41. Rumsey CL, Wedan B, Hauser T, Poinot M (2012) Recent updates to the CFD general notation system (CGNS). In: 50th AIAA aerospace sciences meeting, vol 10, p 6–2012
42. Sethian JA (1999) Fast marching methods. SIAM Rev 41(2):199–235
43. Spalart PR, Allmaras SR (1992) A one-equation turbulence model for aerodynamic flows. AIAA J 94:20
44. Terracol M, Manoha E (2014) Wall-resolved large eddy simulation of a highlift airfoil: detailed flow analysis and noise generation study. In: 20th AIAA/CEAS aeroacoustics conference, AIAA paper 2014-3050
45. Tseng Y-H, Ferziger JH (2003) A ghost-cell immersed boundary method for flow in complex geometry. J Comput Phys 192(2):593–623
46. Vreman AW (1995) Direct and large-eddy simulation of the compressible turbulent mixing layer. Universiteit Twente
47. Zhu WJ, Behrens T, Shen WZ, Sørensen JN (2012) Hybrid immersed boundary method for airfoils with a trailing-edge flap. AIAA J 51(1):30–41