POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

AN IMPLICIT IMMERSED BOUNDARY METHOD FOR INCOMPRESSIBLE NAVIER-STOKES SIMULATIONS

Tesi di Laurea Magistrale in
Aeronautical Engineering - Ingegneria Aeronautica

Simone Cruciani, 966788

Advisor:
Prof. Franco Auteri

Academic year:
2021-2022

Abstract: In this thesis an Immersed Boundary Method (IBM) is developed for a finite-difference DNS code. The IBM approach is introduced in the framework of direction-splitting schemes for the incompressible Navier-Stokes equations and it is based on interpolation and extrapolation techniques. The IB equations are introduced preserving the banded structure of the original problem to allow an efficient solution. Details of the parallel implementation of the solvers and of the IBM are provided. The IB treatment is done implicitly, solving at the same time the governing equations and the discrete-boundary condition equations. This implementation makes this method of interest for simulation of low Reynolds number fluid dynamics problems. The efficacy and the accuracy of the proposed method is tested on both fixed and moving objects with manufactured solutions. An approach to compute the local stress on the IB surface is proposed and it is tested both with manufactured solution and with test cases of a sphere immersed in an uniform flow field.

Key-words: Immersed Boundary method, Finite-differences, Incompressible Navier-Stokes equations

1. Introduction

The growing interest in the study of flows in new and particular fields, such as biological flows [29], particulate flows [18] or natural flyers aerodynamics [32] to cite just a few examples, sets up the challenge for the accurate and efficient numerical simulation of low Reynolds number fluid dynamics in complicated, moving geometries. Typical biological applications that could be of interest are the flow patterns around heart valves [29] or the simulations of inhalation and exhalation of air through the nose, or the flow in micromixers or other micro devices where small deformable objects such as cells are present. In addition to these applications, one research topic with growing interest is the simulation of the flapping wings of insects to possibly emulate it in the design and production of tiny drones. The recent interest in the development of micro air vehicles (MAVs) that can perform indoor or outdoor missions in a D^3 - dull, dirty and dangerous - environment [3], opens the field to the study of low Reynolds number aerodynamics often complicated by fluid-structure interaction (FSI). Design requirements for such vehicles are the small size, the ability to hover and land and take off vertically, the high manoeuvrability also at low speed and in indoor spaces, small noise emissions and autonomy from human piloting [1]. Different design solutions are being explored such as fixed, rotary and flapping wing MAV. Flapping wing is a good choice for MAV design at small size (wing span $\sim 5mm$) in terms of power requirements, maneuverability [1] and flight endurance [20]. The design of small flapping-wing micro air vehicles (FWMAVs) takes inspiration

from insect flight. Since a flapping wing must ensure lift, propulsion and control at the same time, an accurate knowledge of the kinematics and force generation mechanism at low Reynolds number ($\sim 10^2$) is essential. Such a knowledge can be achieved by observation of free and tethered flight of a real insect, for instance the fruit fly *Drosophila melanogaster* [40] [15] [27], by experimental tests [5] and numerical simulations [31]. Studies on insect flight reveal 3D and unsteady aerodynamic phenomena concerning the flapping wing motion such as Wagner's effect related to start and stop motion [1], rapid pitch rotation [11], wake interaction [10] and added mass effects due to accelerations [36]. It must be observed that a stable attachment on the upper wing surface is key for a stable flight [34] [4].

Most of the above studies, either experimental or numerical, analyze an insect-like rigid wing but, considering the thin structure of an insect wing, in recent years simulations start to take into account the effects of wing elasticity on the aerodynamic performance of a flapping wing. Observations with high-speed cameras testify the presence of twist and camber deformations during insect flight [41]. Knowledge of these effects makes it possible to exploits fluid-structure interaction in the design to enhance MAV performance and limit problematic deformations. A coupled approach between aerodynamics, structural dynamics and flight dynamics is needed to design a light and small size FWMAV [33] [34]. Recent studies solved fully 3D coupled FSI problem for a flapping wing introducing a model of the wing structural dynamics. An approach is to use an immersed-boundary flow solver coupled with a structural model made by frames and plates to better simulate the reinforced thin structure of insect wings [38]. This model better captures the real system of venation and membrane and allows one to include non homogeneous, anisotropic properties.

The main challenge in simulating such flows is that, in general, the object moves inside the fluid. Moreover, the need of a coupled fluid-structure-interaction approach, as it could be the flapping wing flight, to fully understand the physics of the problem. Two main different approaches are available to solve problems with a complex and time dependent shape. The first option is to discretize only the fluid domain Ω_f and solve the Navier-Stokes equations in this domain. Arbitrary Lagrangian Eulerian (ALE) methods are an example of this approach. They are based on a mesh that conforms to the boundary $\partial\Omega_f$ at each time step. This method requires a new grid for each time step and interpolation techniques to project the old solution into the new grid.

The other approach is the Fictitious Domain Method (FDM) which consists in extending the problem to a simple shape domain Ω and imposing boundary conditions in a proper way in Ω/Ω_f. Among these methods there is the Immersed Boundary Method (IBM). The IBM was introduced by Peskin [29] but a large variety of versions have been developed thus far. Mittal and Iaccarino [22] classified as IBM all the methods that simulate a flow with embedded boundaries on grids that do not conform to the shape of boundaries. The Cartesian volume grid is built regardless of the surface object grid and it is cut arbitrarily by the immersed boundary. The imposition of boundary conditions becomes the key aspect of IBM and requires an accurate discretization of the governing equations in cells cut by these boundaries. Different variants of IBM have been developed in recent years and they are classified by the way they treat boundary conditions. In [19] a review of the IB methods is available, with a classification for the way each method transfer variables from the fluid grid to the solid one and vice versa in order to satisfy the boundary conditions. There are methods that include a forcing function into the continuous governing equations. This approach is called *continuous forcing approach* and an example is the method of Peskin [29] which includes a Dirac-measure force in the fluid momentum equation to include the effect of the presence of an object. The main idea behind this approach is that it is possible to take into account the presence of an immersed body by means of the forces that the object exerts on the fluid. In such method, the forcing term is spread in the nearby fluid nodes with a discrete delta function. A drawback of such approach is that the spread of the forcing term can blur the boundary interface. The alternative is the so called *discrete forcing approach*. This approach consist of first discretizing the governing equations on a Cartesian grid without regard of the IB and then adjusting the discretization in the cells cut by the IB. This is usually done modifying the computational stencils to directly impose boundary conditions. Such approach treats the boundary as a *sharp interface* [43] and the momentum forcing is applied immediately next to the IB without spreading with a delta function. Such forcing could be introduced implicitly, including the IB velocity inside the linear system to solve, or explicitly by correcting the velocity flow field. The local reconstruction of the corrected velocity field can be done in different ways, examples are [14] that corrects the points with the shortest distance to the IB interface, or [2] that corrects the flow field with a probe along the normal direction. A field extension strategy [45] is typical of such methods to overcome the main drawback which is related to the simulation of moving bodies, when a new fluid point emerges from the IB domain without a significant time history.

An important advantage of the IBM is that the grid generation is simplified, because a Cartesian grid can be used without adapting it to the body shape that could be quite complicated. However a non-conforming, non-aligned grid with respect to the object can be problematic since it is not trivial to impose correctly boundary conditions and control the grid size near the boundaries. This situation could be disadvantageous especially for turbulent flows at high Reynolds, for which the first grid point has to be close enough to the boundary. The other interesting aspect to consider is the application of this method to flows with moving boundaries. IBMs are an attractive alternative to ALE since no new grid has to be generated for each time step and no interpolation

is needed to project the solution on the new grid.
Thanks to this advantage in dealing with moving and possibly deformable objects inside the domain, IBM is a good approach to simulate low Reynolds number flows described above where there is not even the problem of increasing the resolution near the object typical of high Re turbulent flows simulations.
This thesis is organised as follows: in Section 2, the adopted numerical model is presented, mentioning also its parallel implementation. In Section 3, the proposed Immersed Boundary method is described in the case of both moving and still objects. In Section 4, some implementation choices and strategies are detailed. In Section 5, the method used to compute the forces acting on the IB is described and, in Section 6, the numerical results are presented.

2. The incompressible Navier-Stokes solver

The proposed numerical scheme, based on [7], adopts an IBM strategy with a finite-difference solver on a Cartesian fixed grid. Standard centred finite-difference stencils on a non-uniform grid are considered.
The stencils are used for the discrete approximation of the first and second spatial velocity derivatives in the governing equations in grid points far from the boundaries as in Figure 1. Since the expressions are the same in the three directions, only the derivatives along the x axis are shown:

$$\left(\frac{\partial u}{\partial x}\right)_i \approx \left[-\frac{1}{h_{i-1}+h_i}u_{i-1} + \frac{1}{h_{i-1}+h_i}u_{i+1}\right] , \tag{1}$$

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i \approx \left[\frac{2}{h_{i-1}(h_i+h_{i-1})}u_{i-1} - \frac{2}{h_{i-1}h_i}u_i + \frac{2}{h_i(h_i+h_{i-1})}u_{i+1}\right] , \tag{2}$$

where $h_i = x_{i+1} - x_i$ and $h_{i-1} = x_i - x_{i-1}$.

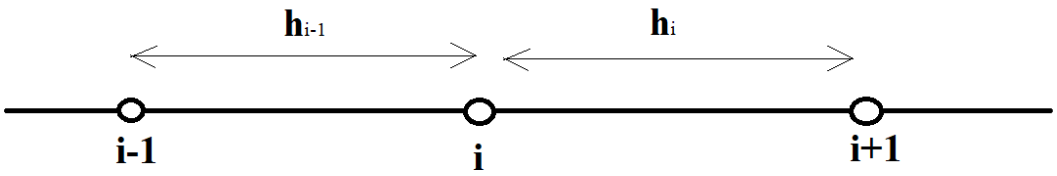


Figure 1: Standard three-point stencil

The first-derivative approximation (1) is second-order accurate in space for any grid whereas the second derivative approximation (2) is second-order accurate only on uniform grids.
A direction-splitting method, introduced by Guermond and Minev [16], is used to uncouple the momentum and continuity equations of the incompressible Navier-Stokes equations allowing a fast and efficient approximation. The main idea of this method is to substitute the Poisson equation of the pressure correction step by a sequence of one-dimensional diffusion equations.
The problem that must be solved consist of:

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2\mathbf{u} , \\ \nabla\cdot\mathbf{u} = 0 , \\ \mathbf{u}(t=0) = u_0 , \\ \mathbf{u}|_{\partial\Omega} = \mathbf{a} , \end{cases} \tag{3}$$

where $\mathbf{u}$ is the fluid velocity, p is the pressure, u_0 is the initial condition and $\mathbf{a}$ the boundary conditions. Both initial and boundary conditions must satisfy compatibility constraints to define a well posed problem.
A common approach to solve it is to use an incremental pressure-correction scheme that can be written as a singular perturbation problem:

$$\begin{cases} \frac{\partial \mathbf{u}_\epsilon}{\partial t} + \mathbf{u}_\epsilon\cdot\nabla\mathbf{u}_\epsilon = -\nabla p_\epsilon + \frac{1}{Re}\nabla^2\mathbf{u}_\epsilon , \\ -\Delta t\nabla^2\phi_\epsilon + \nabla\cdot\mathbf{u}_\epsilon = 0 , \\ \Delta t\frac{\partial p_\epsilon}{\partial t} = \phi_\epsilon - \frac{\chi}{Re}\nabla\cdot\mathbf{u}_\epsilon , \end{cases} \tag{4}$$

with initial and boundary conditions:

$$\begin{cases} \mathbf{u}_\epsilon(t=0) = u_0 \ , \\ \mathbf{u}|_{\partial\Omega} = \mathbf{a} \ , \\ p_\epsilon|(t=0) = p_0 \ , \\ \frac{\partial\phi}{\partial n} = 0 \ , \end{cases} \tag{5}$$

where $\epsilon := \Delta t$ is the perturbation parameter and $\chi \in [0,1]$. The standard incremental scheme is obtained with $\chi = 0$, whereas $\chi = 1$ corresponds to the rotational form. Intermediate values represent a combination of the two schemes.
The formulation of Guermond and Minev [16] is based on the substitution of $-\nabla^2$ in the continuity equation in (4) with a more general operator A. The same stability properties of the original formulation are obtained providing that a bilinear $a(p,q)$ form induced by A, $a(p,q) = \int_\Omega pAq \, d\mathbf{x}$ satisfies these properties:

$$\begin{cases} a(p,q) = a(q,p) \ , \\ ||\nabla q||^2_{L^2} \leq a(q,q) \ . \end{cases} \tag{6}$$

The numerical scheme becomes:

$$\begin{cases} \frac{\partial \mathbf{u}_\epsilon}{\partial t} + \mathbf{u}_\epsilon \cdot \nabla \mathbf{u}_\epsilon = -\nabla p_\epsilon + \frac{1}{Re}\nabla^2 \mathbf{u}_\epsilon \ , \\ \Delta t A \phi_\epsilon + \nabla \cdot \mathbf{u}_\epsilon = 0 \ , \\ \Delta t \frac{\partial p_\epsilon}{\partial t} = \phi_\epsilon - \frac{\chi}{Re}\nabla \cdot \mathbf{u}_\epsilon \ . \end{cases} \tag{7}$$

The definition of the operator A in a proper way allows one to subdivide the three-dimensional problem in a sequence of one-dimensional ones. Guermond and Minev suggested to define the operator $A := (1-\partial^2/\partial x^2)(1-\partial^2/\partial y^2)(1-\partial^2/\partial z^2)$ with appropriate boundary conditions.

$$\begin{cases} \psi - \frac{\partial^2 \psi}{\partial x^2} = -\frac{1}{\Delta t}\nabla \cdot \mathbf{u}, & \frac{\partial \psi}{\partial x} = 0 \ , \\ \varphi - \frac{\partial^2 \varphi}{\partial y^2} = \psi, & \frac{\partial \varphi}{\partial y} = 0 \ , \\ \phi - \frac{\partial^2 \phi}{\partial z^2} = \varphi, & \frac{\partial \phi}{\partial z} = 0 \ . \end{cases} \tag{8}$$

This scheme is a very efficient choice because the solution of the problem involves only tridiagonal linear systems that are quickly solved.

The direction-splitting approach is adopted also for the momentum equation in (7) adopting a Crank-Nicolson scheme for time discretization with the leap-frog strategy for the pressure. The resulting system is:

$$\begin{cases} \frac{\xi^{n+1}-\mathbf{u}^n}{\Delta t} = \frac{1}{Re}\nabla^2 \mathbf{u}^n - \nabla p^{*,n+1/2} - \mathbf{nl}(\mathbf{u}^{*,n+1/2}) \ , \\ \frac{\eta^{n+1}-\xi^{n+1}}{\Delta t} - \frac{1}{2Re}\frac{\partial^2}{\partial x^2}(\eta^{n+1} - \mathbf{u}^n) = 0 \ , \\ \frac{\zeta^{n+1}-\eta^{n+1}}{\Delta t} - \frac{1}{2Re}\frac{\partial^2}{\partial y^2}(\zeta^{n+1} - \mathbf{u}^n) = 0 \ , \\ \frac{\mathbf{u}^{n+1}-\zeta^{n+1}}{\Delta t} - \frac{1}{2Re}\frac{\partial^2}{\partial z^2}(\mathbf{u}^{n+1} - \mathbf{u}^n) = 0 \ , \end{cases} \tag{9}$$

where ξ, η, ζ are temporary variables, $p^{*,n+1/2} = p^{*,n-1/2} + \phi^{n-1/2}$ is the pressure prediction, $\mathbf{u}^{*,n+1/2} = (3\mathbf{u}^n - \mathbf{u}^{n-1})/2$ is a second-order extrapolation of velocity field from previous time steps and **nl** denotes the non-linear term. Once the momentum equation and the pressure step in (7) are solved, the pressure term is updated by: $p^{n+1/2} = p^{n-1/2} + \phi^{n+1/2} - \frac{\chi}{2Re}\nabla \cdot (u^n + u^{n-1})$.
With this approach, the physical diffusion is no more isotropic but the solution diffuses in the computational domain in the three directions in three separated steps. However this scheme does not alter the convergence and the error remains second order [18].
The system of equations (9) can be rewritten in a smart way to optimize the computational phase:

$$\begin{cases} (\eta^{n+1} - \mathbf{u}^n) - \frac{\Delta t}{2Re}\frac{\partial^2}{\partial x^2}(\eta^{n+1} - \mathbf{u}^n) = \Delta t \mathbf{f}^{n+1/2} - \Delta t \nabla p^{*,n+1/2} - \Delta t \mathbf{nl}(\mathbf{u}^{*,n+1/2}) \ , \\ (\zeta^{n+1} - \mathbf{u}^n) - \frac{\Delta t}{2Re}\frac{\partial^2}{\partial y^2}(\zeta^{n+1} - \mathbf{u}^n) = (\eta^{n+1} - \mathbf{u}^n) \ , \\ (\mathbf{u}^{n+1} - \mathbf{u}^n) - \frac{\Delta t}{2Re}\frac{\partial^2}{\partial z^2}(\mathbf{u}^{n+1} - \mathbf{u}^n) = (\zeta^{n+1} - \mathbf{u}^n) \ . \end{cases} \tag{10}$$

With this approach, the unknowns of the problems are no more the sub-step velocity field η, ζ and $\mathbf{u}$, but the system is written considering the velocity differences as unknowns. This approach is computational efficient because it allows one to easily build the *rhs* for the next equation, which is the solution of the previous sub-step, allowing a cheap and fast computation.

Another temporal scheme is included in [7] to allow a simulation with larger time steps by virtue of larger stability margin in terms of CFL condition with respect to Crank-Nicolson. This method, identified by the acronym RK-RAI3, is based on a third-order Runge-Kutta scheme for the convective terms with the subdivision of the time step Δt into three substeps of $\frac{2\Delta t}{\alpha_i}$. The viscous term is treated implicitly, as in Crank-Nicolson scheme, to prevent the severe time step restriction due to explicit treatment ($\Delta t \leq Re\, h^2$). The leapfrog strategy is adopted for the pressure term.

$$\begin{cases} \frac{\alpha_i}{\Delta t}(\eta_i^n - \tilde{\mathbf{u}}_{i-1}^n) - \frac{1}{Re}\frac{\partial^2}{\partial x^2}(\eta_i^n - \tilde{\mathbf{u}}_{i-1}^n) = \mathbf{rhs}\ , \\ \frac{\alpha_i}{\Delta t}(\zeta_i^n - \tilde{\mathbf{u}}_{i-1}^n) - \frac{1}{Re}\frac{\partial^2}{\partial y^2}(\zeta_i^n - \tilde{\mathbf{u}}_{i-1}^n) = (\eta_i^n - \tilde{\mathbf{u}}_{i-1}^n)\ , \\ \frac{\alpha_i}{\Delta t}(\tilde{\mathbf{u}}_i^n - \tilde{\mathbf{u}}_{i-1}^n) - \frac{1}{Re}\frac{\partial^2}{\partial z^2}(\tilde{\mathbf{u}}_i^n - \tilde{\mathbf{u}}_{i-1}^n) = (\zeta_i^n - \tilde{\mathbf{u}}_{i-1}^n)\ , \end{cases} \tag{11}$$

with $\mathbf{rhs} = \frac{2}{Re}\nabla^2\tilde{\mathbf{u}}_{i-1}^n + 2\mathbf{f}_{i-1/2}^n - 2\nabla p_{*,i-1/2}^n - 2\gamma_i(\tilde{\mathbf{u}}_{i-1}^n \cdot \nabla)\tilde{\mathbf{u}}_{i-1}^n - 2\delta_i(\tilde{\mathbf{u}}_{i-2}^n \cdot \nabla)\tilde{\mathbf{u}}_{i-2}^n$. The subscript i stands for the time substep advancement, hence it could be 1,2 or 3, and $\alpha_i, \gamma_i, \delta_i$ are the following coefficients:

$$\alpha_1 = 120/32,\ \alpha_2 = 120/8,\ \alpha_3 = 120/20,$$
$$\gamma_1 = 1,\ \gamma_2 = 25/8,\ \gamma_3 = 45/20,\ \delta_1 = 0,\ \delta_2 = \text{-}17/8,\ \delta_3 = \text{-}25/20.$$

2.1. Modified numerical scheme

The numerical schemes (9) and (11) used for the momentum equation in the incompressible Navier-Stokes equations refer to the splitting scheme proposed by Douglas [12] for the diffusion equation. Indeed, once the pressure gradient ∇p and the convective term $\mathbf{u}\cdot\nabla\mathbf{u}$ are known from the previous time steps, with an appropriate time discretization, the momentum equation in (7) becomes an implicit diffusion equation, such as:

$$\left(1 - \frac{\Delta t}{Re}\nabla^2\right)\frac{\mathbf{u}^{n+1}}{\Delta t} = \frac{\mathbf{u}^n}{\Delta t} + \mathbf{f}^{n+1/2}\,, \tag{12}$$

with $\mathbf{f} = -\nabla p - \mathbf{u}\cdot\nabla\mathbf{u}$. The Douglas scheme in [12] adopts the direction splitting strategy to get:

$$\left(1 - \frac{\Delta t}{2Re}\partial_{xx}\right)\left(1 - \frac{\Delta t}{2Re}\partial_{yy}\right)\left(1 - \frac{\Delta t}{2Re}\partial_{zz}\right)\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \frac{1}{Re}\nabla^2\mathbf{u}^n + \mathbf{f}^{n+1/2}\,, \tag{13}$$

which is an $O(\Delta t^2)$ accurate time discretization of (12), similar to a Crank-Nicolson discretization. The scheme can be rewritten adopting the notation of temporary variables ξ, η and ζ to recover the above schemes (9) and (11). The Douglas scheme has been proved stable in 2D for any geometry domain and in 3D for simple-shaped domain when at least two diffusion operators commute [18]. For the three-dimensional case, in general, no proof of stability is available in the case of non-commutative operators, which is the case of interest in the IB framework. Indeed, for a generic domain which is defined by the shape and the position of the immersed body, such method could become unstable because of the non-commutativity property of the diffusion operators. By extensive numerical testing, it has been noticed that the method becomes unstable solving the diffusion equation considering an immersed object with an arbitrary shape and in a random orientation with respect to the computational grid. The same stability issues have been found by Angot, Keating and Minev [28]. They proposed a modified version of the original Douglas scheme to overcome the limits of this method for this application. The numerical scheme becomes:

$$\begin{cases} \frac{\xi^{n+1}-\mathbf{u}^n}{\Delta t} = \frac{1}{Re}(\partial_{xx}\boxed{\eta^n} + \partial_{yy}\boxed{\zeta^n} + \partial_{zz}\mathbf{u}^n) - \nabla p^{*,n+1/2} - \mathbf{nl}(\mathbf{u}^{*,n+1/2})\ , \\ \frac{\eta^{n+1}-\xi^{n+1}}{\Delta t} - \frac{1}{2Re}\frac{\partial^2}{\partial x^2}(\eta^{n+1} - \boxed{\eta^n}) = 0\ , \\ \frac{\zeta^{n+1}-\eta^{n+1}}{\Delta t} - \frac{1}{2Re}\frac{\partial^2}{\partial y^2}(\zeta^{n+1} - \boxed{\zeta^n}) = 0\ , \\ \frac{\mathbf{u}^{n+1}-\zeta^{n+1}}{\Delta t} - \frac{1}{2Re}\frac{\partial^2}{\partial z^2}(\mathbf{u}^{n+1} - \mathbf{u}^n) = 0\ . \end{cases} \tag{14}$$

The $\boxed{\text{boxed}}$ terms are the alterations with respect to the original version, which could be recovered by substituting $\mathbf{u}^n$ in the place of $\boxed{\eta^n}$ and $\boxed{\zeta^n}$.
The modified scheme remains a Crank-Nicolson discretization of the diffusion equation and preserves the second order consistency in time. This is provable by eliminating all the temporary variables in (14) with a combination of the set of equations above (all the steps are explained in [28]). The scheme simplifies to:

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \frac{1}{Re}\nabla^2\left(\frac{\mathbf{u}^{n+1} + \mathbf{u}^n}{2}\right) + \mathbf{f}^{n+1/2} - \frac{\Delta t^2}{2Re^2}(\partial_{xxyy} + \partial_{xxzz} + \partial_{yyzz})\left(\frac{\mathbf{u}^{n+1} + \mathbf{u}^{n-1}}{2\Delta t}\right) + O(\Delta t^4)\ . \tag{15}$$

The changes in the modified version are related to the fact that the original scheme uses the final step velocity $\mathbf{u}^n$ in all the three explicit diffusion operators: ∂^n_{xx}, ∂^n_{yy} and ∂^n_{zz}. However, it must be considered that $\mathbf{u}^n$ is, in general, smooth only in the z direction. Indeed, since the x and y direction are not treated implicitly in the final velocity computation, as done for the z direction, $\mathbf{u}^n$ is not guaranteed to have the same discrete regularity in x and in y as in z. These changes allow one to match the explicit diffusion operator at time level n with the sub-step field that is obtained solving the problem in that respective direction, along which the field is smooth. Indeed, the diffusion operators in the x direction, ∂_{xx}, is applied to η^n, whereas ∂_{yy} is used for the sub-step field ζ^n.
The modified scheme is less efficient with respect to the original one because of the need to store the sub-step velocity field η and ζ at each time step. Moreover, the *rhs* for each one-directional problem is no more the solution for the previous directional problem, as in the original version in (11) and (10), and it must be computed each time after the solution of each directional problem. This is caused by the fact that it is no more possible to write the problem considering the velocity differences as unknowns. No proof of unconditional stability is available for the modified scheme but the numerical simulations in [18] and [28] suggests that this scheme is always stable for the diffusion equation. For such reason the modified version of the Douglas scheme is adopted in this thesis.

2.2. Parallel implementation

The algorithm is implemented in parallel [7]. The computational domain is partitioned with a Cartesian block decomposition (Figure 2). Each processor operates on a single block and communication is insured with the Message Passing Interface library.

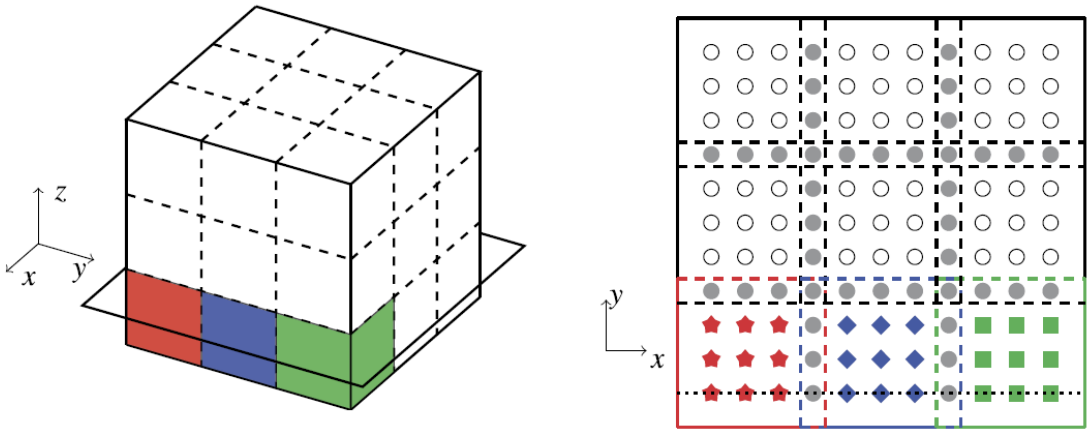


Figure 2: Cartesian block decomposition. Figure taken from [7]

The grid is divided into internal points x_i and shared interface points x_s. Internal unknowns are shown in Figure 2 with red, blue and green points whereas the grey dots are the shared interfaces. For internal points, the assembly of the right-hand side does not need communication which however is needed for the shared interface points. This method keeps the communication among processors to a minimum, indeed it is only needed for blocks that share an interface.

2.2.1 Schur complement method

The solution of tridiagonal systems in parallel is done with the Schur complement method. The original tridiagonal system $A\,u = f$ is rewritten considering the subdivision between internal and shared unknowns. The reordered system becomes:

$$\begin{bmatrix} A_{ii} & A_{is} \\ A_{si} & A_{ss} \end{bmatrix} \begin{bmatrix} u_i \\ u_s \end{bmatrix} = \begin{bmatrix} f_i \\ f_s \end{bmatrix},$$

where the subscript i stands for internal unknowns and s stands for shared ones.
The reordered system is no more tridiagonal. Indeed A_{ii} is tridiagonal and A_{ss} is diagonal but A_{si} and A_{is} are sparse matrices. Nevertheless it can be reduced to a triangular form using the block Gaussian elimination. Therefore the solution is obtained solving the following linear systems:

$$Su_s \;=\; (A_{ss} - A_{si}A_{ii}^{-1}A_{is})u_s \;=\; f_s - A_{si}A_{ii}^{-1}f_i \;, \tag{16}$$

$$A_{ii}u_i \;=\; f_i - A_{is}u_s \;, \tag{17}$$

where S is the Schur complement. This approach decouples the solution for the shared interface unknowns and the internal ones.
The system (16) gives the equations for the shared unknowns and its solution is needed in the computation of the internal unknowns in the equations (17). The key property of this approach is that the Schur complement preserves the banded structure of the original system. Therefore, the systems that have to be solved remain tridiagonal also in parallel.
The systems (16) and (17) are solved by a sequence of steps:
Preprocessing:

- Assembly of the Schur complement $S \;=\; (A_{ss} - A_{si}A_{ii}^{-1}A_{is})$

Runtime:

- Computation of right-hand side for the internal points f_i
- Computation of the processor's contribution to the right-hand side of the equation for the shared interface unknowns $f_s - A_{si}A_{ii}^{-1}f_i$
- $*$ Assembly by communication of $f_s - A_{si}A_{ii}^{-1}f_i$
- Solution for the shared interface unknowns u_s
- Computation of the right-hand side for the internal points $f_i - A_{is}u_s$
- Solution for the internal unknowns

This method minimizes the communication between processors which is needed only in the assembly of the right-hand side $f_s - A_{si}A_{ii}^{-1}f_i$, identified by symbol $*$ in the algorithm. Once the system (16) has been assembled, it is solved by each processor by virtue of the small size of the Schur complement with respect to that of the system of internal unknowns u_i (in general $N_s \;<<\; N_i$), avoiding a successive communication among all the processors.
However, considering the IB framework and its potentiality with moving and flexible objects, the code is implemented with the assembly of the Schur complement done in the runtime phase, with additional communication with respect to [7]. Indeed, the relative position between the IB and the computational grid could change at each timestep requiring to adapt the matrices for each time advancement.

3. Immersed boundary method

The solver used in this paper is based on an immersed boundary method. A fixed Eulerian grid is generated regardless of the presence of the solid object which is immersed in this volume and cuts randomly the computational grid. Since the grid does not conform to the boundary surface, it is not trivial to correctly impose boundary conditions on the solid body. In this section, the treatment of the IB is described considering also the proposed approach in case of a moving object.
There are several IBM implementations available, each one different from the others by the treatment of the immersed body. In [35], the methods are divided by the way the IB is introduced. In particular, a classification is done between *diffuse* and *sharp* interface methods. In the first group there are all methods which use a discrete delta function to distribute the IB forcing term on the Eulerian grid nodes. An example is Peskin's method [29] or, as alternative, [39] which differs from the former in the way the forcing term is computed. In [39], the computation of a body force is done on the Lagrangian marker uniformly distributed on the immersed surface and, then, the intermediate velocity is corrected with the spreading of such body force on the Eulerian grid. As alternative treatment, the *sharp interface* methods are introduced to overcome the smoothness of the boundary which occurs by the spreading effect of the discrete delta functions. A large variety of methods can be collected in this group. Examples are the *cut-cell* methods which are based on a Cartesian grid remodeled in the proximity of the object by the IB geometry. Furthermore, a possibility is to use the *immersed interface* methods [42] that correct the finite difference stencils near the interface introducing jump conditions, or the *hybrid Cartesian-immersed boundary* schemes [23] that reconstruct the velocity field by satisfying the boundary condition. The reconstruction of the velocity field can be done in several ways, with linear [14], bilinear [44] or wall-normal [2] interpolations, usually done, in the fractional step framework, for the intermediate velocity before the solution of the continuity equation. Such methods use an explicit approach to deal with the presence of the IB and to impose boundary conditions. With an explicit scheme, the momentum equation in the incompressible Navier-Stokes model is solved regardless of the presence of the IB, then, before the pressure step, a correction is introduced in the grid points around the body to accurately impose a velocity that satisfies the boundary conditions on the immersed surface. However, the explicit method struggles in simulations at low Reynolds number where the required time step becomes too small. Indeed, even if the explicit correction ensures a proper

velocity around each immersed body, the lower the Re is, the faster the diffusion term spreads that value in the solution. This phenomenon leads to a severe limit on the time step that is needed to accurately simulate such flows. For very low Reynolds numbers, this approach could become unfeasible. In [9], this phenomenon is highlighted and the comparison of a semi-implicit treatment of IB with a fully explicit one demonstrates that the integration could be performed with a CFL significantly higher preserving the same accuracy. The implicit treatment solves these issues. With this approach, the boundary condition equations are discretized and solved at the same time of the momentum equation.

The approach here proposed is based on an implicit treatment of the immersed boundary. The imposition of the boundary condition is done modifying the discrete stencils in the proximity of the IB. In particular, for a grid point close to the immersed boundary, a linear equation is introduced to take into account the boundary conditions. Linear equations that result from the discretisation of the boundary condition are then solved at the same time of the discrete governing equations causing the boundary conditions to be implicitly imposed.

3.1. IB boundary condition

Near the interface between the fluid and the object, it is possible that one of the three points of the standard stencil is outside Ω_f. In this case, it is necessary to consider the presence of the solid body, impose the Dirichlet boundary condition and adapt the grid around the boundary. This process is done separately for each of the three one-dimensional equations involved in the solution of the momentum equation.
Let us consider the situation of Figure 1 and let, for example, x_{i-1} be a "ghost point" outside Ω_f and inside the solid, as in Figure 3, and define the intersection point of an x-aligned straight line passing through x_{i-1} with the object boundary x_{bL}. u_{bL} is the known velocity in this point and $\gamma_L = x_i - x_{bL}$ as in Figure (3). If no boundary fitted stencil is used, the boundary condition will be imposed in x_{i-1} resulting in a loss of accuracy.

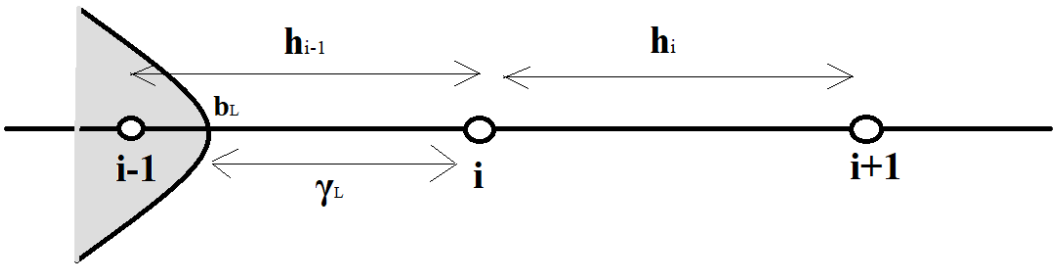


Figure 3: Left boundary intersection

There are several possibilities to impose the boundary condition, the first one is to add a new grid point on $\partial\Omega_f$ and modify the stencil to use this point. However, in the proposed scheme, no grid points are added and the fixed Eulerian grid remains the same for each time step. The proposed approach consists in deriving a linear equation that approximates the presence of the IB and this can be done in several ways. An option is to linearly extrapolate the solution in the first point outside the fluid domain taking the presence of the immersed boundary into account defining a value for the "ghost point" $i-1$:

$$u_{bL} = \frac{\gamma_L}{h_{i-1}} u_{i-1} + \left(1 - \frac{\gamma_L}{h_{i-1}}\right) u_i \ . \tag{18}$$

In this "ghost point", the governing equation is not solved and it is substituted by (18). However, numerical problems could arise in situations where the boundary points are very close to the fluid grid points, with $\gamma_L \approx 0$. In these situations, a linear extrapolation can present a significant overshoot and, if such value overcomes the maximum velocity in the computational domain, this can generate numerical issues in terms of reduction of the maximum time step because of the CFL limit.
Another option for the linear extrapolation (18) is possible, considering an extrapolation from a farther point from the boundary. This could guarantee improvements in terms of stability according to [28] because overshoot problems are limited, resulting in:

$$u_{bL} = \frac{\gamma_L + h_{i+1}}{h_i + h_{i+1}} u_{i-1} + \frac{h_i - \gamma_L}{h_i + h_{i+1}} u_{i+1} \ . \tag{19}$$

However, using this linear extrapolation the matrix to solve is no more tridiagonal, because the equation for the point $i-1$ involves the point $i+1$, and this is not optimal for an efficient solver.

In general, the numerical problems described above are due to the fact that a linear equation is always written for the first point outside Ω_f. It is possible to employ a linear interpolation for the point inside Ω_f closest to the immersed boundary, since the overshoot never arises with a linear interpolation. In this case, the linear equation resulting from the discretization of the boundary condition defines implicitly a value for the fluid point closest to the boundary (point i in Figure 3):

$$u_{bL} = \left(\frac{\gamma_L}{h_i} + 1\right) u_i - \frac{\gamma_L}{h_i} u_{i+1} \ . \tag{20}$$

With this interpolation, no governing equation is solved at the first grid point in Ω_f and a linear velocity profile is imposed in the proximity of the immersed boundary. This approach is reasonable for laminar flows at low Re, whereas it needs a grid fine enough near the wall for flows at high Re.
So, to overcome instability issues, the imposition of boundary conditions can be always done using (20) and a tolerance is added to consider a point slightly inside the body as a fluid point, to keep the method as robust as possible. In case of a still body, the scheme maintains the overall second-order spatial accuracy because the discrete derivative stencils (1) and (2) and the linear interpolation for the boundary condition (20) are second order accurate in space.
The accuracy of these stencils is numerically tested in a one-dimensional heat equation and the second-order spatial accuracy is confirmed (Section 6.1.1).

3.2. Moving Immersed Boundary

In case of moving boundaries, particular attention must be paid for situations where a new grid point emerges in the fluid domain as in Figure 4. In general, the velocity and pressure values are not meaningful in the points outside the fluid domain and discontinuities across the boundary interface can generate numerical oscillations. Indeed, when a new "fresh" point enters the fluid domain it has not a significant time history and this can lead to instabilities when the governing equations are solved in this grid point.
To overcome this issue, the following scheme is proposed. The imposition of boundary conditions is done implicitly in the fluid point closest to the interface thanks to the linear interpolation (20) but, at the same time, the linear extrapolation (18) is performed. The extrapolation of fluid variables inside the solid is a typical feature of such methods, examples of this approach are [18, 44].

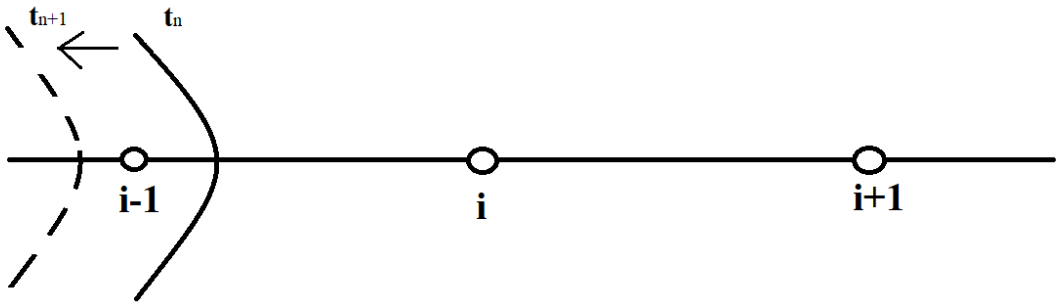


Figure 4: Moving Immersed Boundary: New fluid point. At time step t^n the points i and $i+1$ are fluid points whereas the point $i-1$ is inside the IB. At the successive time step, t^{n+1}, the point $i-1$ crosses the IB surface and emerges into the fluid

The implicit interpolation allows one to always assign a meaningful value to all new points emerged into the fluids because in their computation there is no need of previous time steps values. Therefore, in situations as in Figure 4, the point $i-1$ becomes a new fluid point at the time step t^{n+1} and the linear interpolation to impose the boundary condition defines the value of velocity ($\mathbf{u}_{i-1}^{n+1}$) at this time step. However, at the same time step t^{n+1}, in the point i the discrete governing equations are solved and thus second velocity derivative must be correctly defined for both time steps t^n and t^{n+1}. Indeed the time discretization approximates ∂_{xx} as $0.5(\partial_{xx}^n + \partial_{xx}^{n+1})$. To properly define ∂_{xx}^n with a discrete three-point stencil, a meaningful value for velocity in the point $i-1$ is needed at the time step t^n even if it is inside the solid. The linear extrapolation ensures that the point inside the immersed surface keeps track of the IB boundary condition, guarantees a significant velocity value for that point and allows one to correctly define a boundary fitted second spatial derivative for velocity for each time step.
In particular, it is possible to substitute the extrapolation (18) into the standard centred scheme for the discrete second derivative (2) to get the boundary fitted second derivative stencil:

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i \approx \left[\frac{2}{\gamma_L(h_{i-1}+h_i)}u_{bL} - \frac{2}{h_{i-1}+h_i}\left(\frac{1}{\gamma_L}+\frac{1}{h_i}\right)u_i + \frac{2}{h_i(h_{i-1}+h_i)}u_{i+1}\right] . \tag{21}$$

In this way the proposed scheme is able to prevent oscillations also with moving boundaries and to maintain the second order accuracy because both the linear extrapolation and the centred finite-difference scheme are second order.
The above stencils are written for the left-boundary intersection. In order to generalize and complete the discussion, finite-difference approximations for the right intersections are also included.

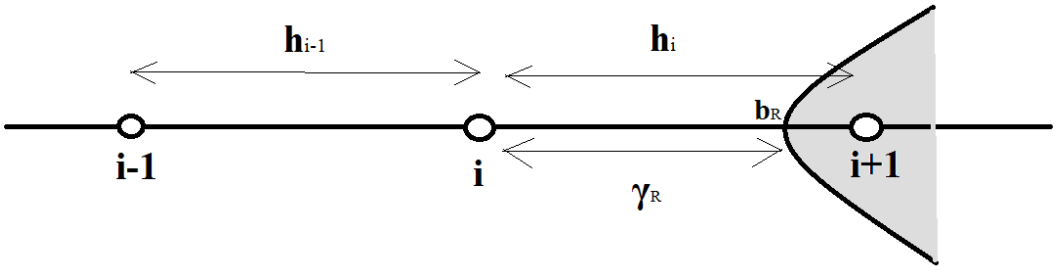


Figure 5: Right boundary intersection

For the right intersection in Figure 5, it is possible to use a linear interpolation between the points x_{i-1} and x_{bR} to compute u_i and implicitly enforce the boundary condition as:

$$u_{bR} = -\frac{\gamma_R}{h_{i-1}}u_{i-1} + \left(1+\frac{\gamma_R}{h_{i-1}}\right)u_i . \tag{22}$$

The extrapolation for the first solid point ($i+1$ in Figure 5) is:

$$u_{bR} = \left(1-\frac{\gamma_R}{h_i}\right)u_i + \frac{\gamma_R}{h_i}u_{i+1} . \tag{23}$$

It is possible to substitute the extrapolation (23) into the standard three point second derivative stencil (2) to obtain:

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_i \approx \left[\frac{2}{h_{i-1}(h_{i-1}+h_i)}u_{i-1} - \frac{2}{h_{i-1}+h_i}\left(\frac{1}{\gamma_R}+\frac{1}{h_{i-1}}\right)u_i + \frac{2}{\gamma_R(h_{i-1}+h_i)}u_{bR}\right] . \tag{24}$$

The accuracy of this approach is tested in Section 6.1.1 with a one-dimensional heat equation with a moving boundary. As stated previously, this approach is able to maintain the second-order accuracy and prevents numerical instabilities also with a moving body, which is of primary interest to simulate fluid-structure interaction problems. The combination of interpolation and extrapolation allows one to correctly treat the discontinuity across the fluid-solid interface that is a source of oscillations in the fluid domain in case of a moving body. Furthermore, this IB treatment does not introduce severe time-step limitations which can slow down the simulations. Indeed, the time step should not exceed the time interval necessary for the IB to move one spatial interval.

3.3. Boundary fitted stencils

The boundary condition on the IB surface is enforced adopting the strategy presented in Section 3.1. However, it must be considered that the grid points that are outside the fluid domain Ω_f have no meaningful values for pressure and velocity. In general, such points can enter into the three-point stencils used to discretize each term in (9) and (8). Indeed, this is the case of the Eulerian grid point which are very close to the IB. Both the velocity and the pressure fields are not guaranteed to be regular on the grid line which crosses the IB surface and only continuity is ensured. This aspect requires a special treatment and modifications of the stencils are used to correctly define each term of the Navier-Stokes equations. In [18], a strategy to correct the *rhs* terms in the points closest to the IB is proposed and new boundary fitted stencils for the diffusive term $\nabla^2\mathbf{u}$, the pressure gradient ∇p and the divergence of the velocity $\nabla\cdot\mathbf{u}$ are defined. On the opposite, the advection term $(\mathbf{u}\cdot\nabla)\mathbf{u}$ no requires fitting [18], because its contribution in the point closest to the surface can be assumed negligible, even if it is still possible to derive a boundary fitted stencil also for such term.

Moreover, such boundary fitted stencils are a generalization of the boundary fitted derivatives presented above to extend the treatment for the 1D heat equation in the Navier-Stokes equations with moving objects.

The Laplacian term is modified adopting the same strategy presented in Section 3.2. Indeed, a linear extrapolation is used to explicitly assume a correction for the first grid point inside the solid body. In this way, the standard three-point second derivative stencil (2) is modified and the corrected version (21) and (24) is used to correct the diffusive term. In this framework, the three-point stencil uses only two fluid points and the boundary condition on the immersed boundary.

The pressure gradient term is modified in a similar way. Let us consider the standard situation in Figure 1, the standard stencil is:

$$\left(\frac{\partial p}{\partial x}\right)_i = \frac{p_{i+1} - p_{i-1}}{h_{i-1} + h_i} , \tag{25}$$

where $h_i = x_{i+1} - x_i$ and $h_{i-1} = x_i - x_{i-1}$. An explicit correction is used to modify the first solid grid point value. Linear or constant extrapolations are possible. However, in this approach, the strategy adopted consists of a constant extrapolation of the pressure field past the IB interface. In particular, let us consider a *left* boundary intersection, as in Figure 3, the pressure gradient for the grid point i is computed adopting the constant extrapolation for the point $i-1$ which is inside the IB: $p_{i-1} = p_i$. The same strategy is used for *right* intersections and for the x, y and z direction.

The divergence term is also modified. Let us consider the standard situation in Figure 1, this term is discretized as:

$$(\nabla \cdot \mathbf{u})_{i,j,k} = \frac{u_{i+1,j,k} - u_{i-1,j,k}}{x_{i+1} - x_{i-1}} + \frac{v_{i,j+1,k} - v_{i,j-1,k}}{y_{j+1} - y_{j-1}} + \frac{w_{i,j,k+1} - w_{i,j,k-1}}{z_{k+1} - z_{k-1}} , \tag{26}$$

where $\mathbf{u} = (u, v, w)$. Let us consider only a *left* intersection along the x direction (Figure 3), the x derivative term becomes:

$$(\nabla \cdot \mathbf{u})_{i,j,k} = \frac{u_{i+1,j,k} - u_b}{x_{i+1} - x_b} + \frac{v_{i,j+1,k} - v_{i,j-1,k}}{y_{j+1} - y_{j-1}} + \frac{w_{i,j,k+1} - w_{i,j,k-1}}{z_{k+1} - z_{k-1}} . \tag{27}$$

The same strategy is adopted for the other directions and for the *right* intersections.

4. IB implementation

The Immersed Boundary method described above is implemented in the Navier-Stokes solver developed by *Chiarini et al.* [7], coded in *Fortran*. It is based on the direction-splitting approach on co-located grids presented above and is implemented in parallel using the MPI library.
The adopted finite difference discretization coupled with a three-point centred stencil leads to matrices that are three-diagonal. As explained above, in the parallel implementation with the Schur complement method [7] and the block Gaussian elimination, the system is reordered and preserves the tridiagonal structure.
The presence of the immersed boundary, with an implicit treatment, modifies the coefficients of the systems. Indeed, the equations needed to impose the boundary conditions depend on the relative position between the immersed surface and the computational grid, which is not constant if the body is moving inside the domain. Nevertheless, the key aspect of the immersed boundary technique described above is that the structure of each matrix is unchanged and always tridiagonal systems must be solved. These aspects allow us to use the same tridiagonal linear solver used in the original method. The Thomas algorithm has been used to solve the linear systems.

4.1. Thomas algorithm

The most common method to solve tridiagonal systems is the Thomas algorithm. It is a method that allows one to solve a $N \times N$ linear system with a low computational cost of the order of N *flops*. Indeed, it exploits the band structure of the matrix to keep the number of operations to a minimum. There are other methods to solve a tridiagonal linear system, though. For example, without the presence of an IB or at least with a fixed IB, the most efficient approach is to factor the matrix once and for all at the beginning of the computation. This approach exploits a LU or LDU factorization [30]. Considering a problem of dimension $N \times N$ the factorization procedure has a computational cost of $\approx$ *4N flops*. The forward and backward substitutions require $\approx$ *3N flops*. In general, this implementation increases a bit the computational cost of the algorithm with respect to the standard Thomas algorithm but, considering that the factorization is not repeated for each time steps, the

total cost is reduced. Moreover, an efficient way to implement this algorithm is to invert the diagonal array after the factorization to avoid the division operations in the part of the algorithm that is repeated for each time step, increasing the computational performance of the solver [30]. Indeed, the division operations are expensive because they take more clock cycles than additions and multiplications [13]. In the case of a moving IB, the problem becomes quite different. In particular, the relative position of the computational grid and the body surface changes for each time steps leading to different matrices to solve. In this case, it is not convenient to factor the matrix at every time step increasing the computational cost without a real advantage. There is no point in using factorisation on the processors that do not deal with the IB, since this will negatively affect the load balance between processors.
A reasonable strategy is to use the same solver for all blocks. The adopted algorithm is based on a standard Thomas algorithm without factorization that costs $\approx$ *5N flops* and allows one to gain something in terms of computational cost with respect to the previous implementation with the LDU factorization when the linear system to be solved changes at every time step.

Consider a generic problem $A\,x = f$ with $A \in R^{N \times N}$ and $x, f \in R^N$ such that $x = \{x_1, ..., x_N\}$, $f = \{f_1, ..., f_N\}$ and A is the tridiagonal matrix:

$$\mathrm{A}=\begin{bmatrix} b_1 & c_1 & 0 & \dots & \dots & \dots & 0 \\ a_1 & b_2 & c_2 & 0 & \dots & \dots & 0 \\ 0 & a_2 & b_3 & c_3 & 0 & \dots & 0 \\ 0 & & \ddots & \ddots & \ddots & & 0 \\ 0 & & & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & a_{N-2} & b_{N-1} & c_{N-1} \\ 0 & \dots & \dots & \dots & 0 & a_{N-1} & b_N \end{bmatrix},$$

The implemented Thomas algorithm is :

for $i = 2, ..., N$ **do**
$w = a_{i-1}/b_{i-1}$
$b_i := b_i - w\, c_{i-1}$
$f_i := f_i - w\, f_{i-1}$
end for

followed by the back substitution :

$x_N = f_N/b_N$

for $i = N-1, ..., 1$ **do**
$x_i = (f_i - c_i x_{i+1}) \,/\, b_i$
end for

Up to this point only the non-periodic version of Thomas algorithm has been considered. It is possible to derive an implementation for the periodic case to keep the implementation as general as possible.

4.2. Data Structure

In the preprocessing phase of the immersed boundary treatment, a specific data structure is built to compute and store all the information that is needed to deal with the presence of the immersed body.
The input data provides the position and the discretization of the body surface. In the parallel implementation, the immersed surface data is shared among all the processor through the MPI library. Each processor has the information of all the immersed surface even if it is not inside its block.

4.2.1 Immersed Boundary Data Structure

The first step to deal with the IB inside the computational domain is to compute all the intersection points between the Cartesian computational grid and the surface triangulation of each immersed object. The intersections are the key variables to treat the presence of the object into the fluid. A subroutine takes care of scanning all the faces of the immersed boundary, verifying if they intersect a grid line for each direction *x,y* or *z*, and computing the intersection points, also defining the points that will be used in the interpolation and extrapolation phase. This process involves the choice of the point nearest to the intersection and check if it is a fluid point or not. Indeed, boundary conditions are enforced in the fluid point nearest to the intersection, so only fluid points must be involved in this step.

The intersection points are found exploiting Ray-Triangle algorithms [24] and the barycentric coordinate system (see Section 4.2.2) is used to describe each triangular face. The grid-boundary intersection routine is one of the most critical aspects. Indeed, the algorithm that identifies and computes the intersection of a ray with a triangular face has to be robust. In general, some algorithms may struggle in situations where a ray intersects an edge or a vertex and, for numerical issues, the intersection could not been found. By extensive numerical testing it has been observed that this phenomenon can trigger instabilities of the IB method. To prevent this situation, the algorithm has been adjusted with some tolerances on the criteria that select if a point is inside the triangle or not.
Once the intersections have been found for each immersed surface (more than one object can be taken into account) and for all the three directions x, y and z, it is important to precisely define in which point the boundary condition equation is solved instead of the governing equation, what points enter the interpolation and extrapolation procedures, the relative position between the body and the grid and which face of the discrete surface grid is cut by the Cartesian grid.
This process must be done for all the three directions x, y and z because the solution process involves a sequence of three one-dimensional cases. The boundary conditions on the IB are indeed enforced for each direction in each of the one-dimensional equations of the system that discretize the momentum equation in (9) or (11).

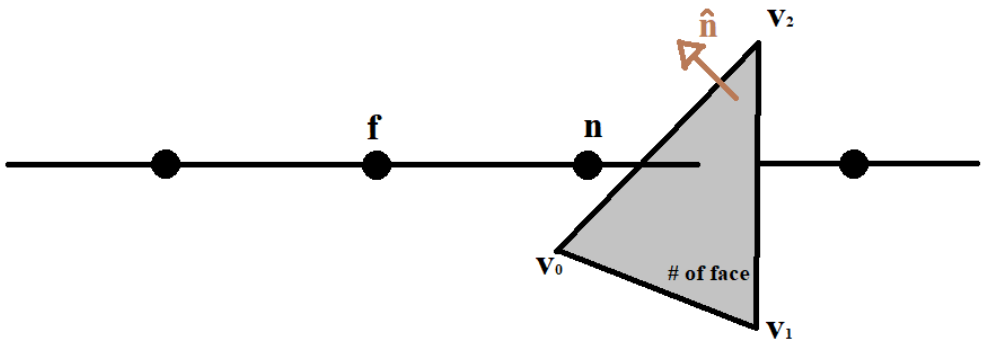


Figure 6: Elements of IB Data Structure: n and f are used in the linear interpolation with the intersection point on the IB triangular face. $\hat{n}$ is the normal versor pointing outwards the immersed surface.

A routine is involved in the collection of all the necessary data and in the assembly of the IB data in a dedicated type variable. The information is stored in the *ImmersedBoundaryData* structure that contains all the data explained in Tables 1 and 2 and some of them are shown in Figure 6

Name	Dimension
points_n	nSurf × nGrids × dim
points_f	nSurf × nGrids × dim
delta	nSurf × nGrids × dim
delta_extr	nSurf × nGrids × dim
x	nSurf × nGrids × dim
y	nSurf × nGrids × dim
z	nSurf × nGrids × dim
n_triangle	nSurf × nGrids × dim
coeff_interp	nSurf × nGrids × dim

Table 1: Immersed Boundary data structure: Name and dimension for each variable

The parameters **dim**, **nSurf** and **nGrids** are read from the input file. In particular **dim** defines the size of the problem (for a 3D problem **dim**= 3), **nSurf** represents the number of immersed surfaces (more than one is possible) and **nGrids** represents the number of computational grids (equal to 1 for the co-located solver adopted in this paper). Each variable is a pointer to an array of different size which depends on the number of intersection points found for each direction. The number of intersection points is saved in **npoints** and used to allocate only the required memory space. For each direction, the variable **points_n** points to an array of

integers which contains the indices of the point which identifies the governing equation that is substituted by the linear interpolation (Figure 6). The variable **points_f** stores the indices of the other point that enters into the implicit interpolation procedure (Figure 6). The variables **delta** and **delta_extr** point to arrays of real numbers which define the coefficients needed in the interpolation and extrapolation procedure. The variables **x**, **y** and **z** stores the physical coordinates of the intersection point. This point belongs to one triangular face of the immersed-surface triangulation. The index of this face is stored in the array pointed by **n_triangle**. Once the intersection point and the face are found, the three interpolation coefficient of the barycentric coordinate system are stored in **coeff_interp** to retrieve the correct boundary condition in the intersection point (details in Section 4.2.2).

Name	Description
points_n	Indices of BCs application points
points_f	Indices of points needed in the interpolation
delta	Coefficient to build interpolation
delta_extr	Coefficient to build extrapolation
x	Physical coordinate of the intersection point
y	Physical coordinate of the intersection point
z	Physical coordinate of the intersection point
n_triangle	Index of the face cut by Cartesian grid
coeff_interp	Barycentric coordinates of the intersection point

Table 2: Immersed Boundary data structure: Name and a brief description for each variable

4.2.2 Barycentric coordinate system

The position of the intersection point on each triangle surface defines a set of three interpolation coefficients. These coefficients are the coordinates of a point in the local barycentric coordinate system and they are used to verify if that point is inside that triangle [24] and then to retrieve the value of boundary velocity in a point that is, in general, different from the discretization ones of the immersed surface. The boundary velocity is explicitly available only in the discretization points that define the IB and not in all the surface. A linear interpolation is used to compute an accurate estimation of the boundary velocity in each intersection point with the three values of velocity available in the vertices of each triangular face.
The barycentric coordinate system is used to retrieve this information. Consider a set of three points $\mathbf{v}_0, \mathbf{v}_1$ and $\mathbf{v}_2$ that are the vertices of a triangle $\mathbf{T}$. Let $\mathbf{q} \in \mathbf{T}$ be a point inside that triangle, it is possible to define an unique set of three coefficients k_0, k_1 and k_3 such that:

$$\mathbf{q} = k_0\mathbf{v}_0 + k_1\mathbf{v}_1 + k_2\mathbf{v}_2 \ , \tag{28}$$

that satisfies these properties:

$$k_0, k_1, k_2 \in [0, 1] \ , \tag{29}$$

$$k_0 + k_1 + k_2 = 1 \ , \tag{30}$$

When $\mathbf{q}$ is the intersection point of a grid line with the immersed surface, it is possible to build the barycentric coordinate system computing the set of coefficients and use this reference frame to compute the correct boundary condition. Let $\mathbf{U}_0, \mathbf{U}_1$ and $\mathbf{U}_2$ be the boundary velocity in the vertices of the triangle, it is possible to compute the boundary condition in $\mathbf{q}$ as :

$$\mathbf{U}_q = k_0\mathbf{U}_0 + k_1\mathbf{U}_1 + k_2\mathbf{U}_2 \ . \tag{31}$$

4.3. IB implementation strategy

In this part, some details about the proposed IB method are provided. In the first stages of the simulation, once all the input data are read, the code starts with the initialization phase. In this part, once the time scheme is selected, all the matrices that discretize the system of equations (8) and (9) with the finite difference approach are built, without taking into account the presence of the IB. In the code version without the presence of IB, all the matrices are factored, allowing a more efficient computation. In case of IB, the factorization is not carried out because, with a moving IB, the factorization would be required for each time step, losing any advantage, as

explained in Section 4.1.

Once the initialization phase is finished, the code processes the input data file of the IB discretization and computes each intersection between the Eulerian grid and the surface faces. In this stage, the IB data structure described in Section 4.2.1 is built. When all the IB data are available, the phase in which the implicit IB method is applied begins.

In this part, the strategy adopted in the implementation is to temporarily copy all the initialized original matrices and then, for each direction, modify the coefficients where the boundary condition is applied. In this part, the indices of **points_n** linked with the points of application of the boundary conditions on the IB are recovered and they are used to select which discrete governing equation must be substituted by the discrete version of boundary condition (20) or (22). In this stage the coefficients are modified to ensure the correct discrete equation. In this phase, the right-hand side must be modified too, paying particular attention to the parallel implementation. The discrete linear boundary condition equation to enforce, (20) or (22), requires the *rhs* to be adjusted. In particular, the *rhs* is simply the IB boundary speed $\mathbf{U}_b$, which could be an assigned value or recovered by the velocity of the surface triangulation points thanks to the barycentrical coordinate system, as described in Section 4.2.2.

Parallel implementation In the case of parallel computing, there could be situations that need to be treated in a special way. Each processor is able to identify all the intersections that are in the respective block, included the case when the intersection point is on the interface. If the **points_n** is a local point and not shared, there are no differences with respect to the serial case. The exception is when this point is immediately adjacent to the interface. Indeed, in this situation, the boundary condition slightly modifies also the coefficients for the *shared* unknowns in the Schur complement.

The other possible case is that **points_n** is exactly the shared interface point. In this situation both processors are able to identify and collect this intersection and both processors need to modify their respective contribution to the Schur complement and their local matrices to correctly impose the boundary condition. In this case, the reconstruction of the *rhs* becomes no more trivial because the *rhs* must be computed for the shared unknowns and not for the internal one. In particular, the same approach described in [7] must be used to be coherent with that parallel treatment of the *rhs*. In particular, the most important aspect is to implement such term in the shared interfaces in such a way that the boundary condition is considered only once. Indeed, the adopted strategy for the parallel treatment of the *rhs*, described in [7], exploits the linearity of the problem to split the *rhs* in the adjacent processors by partially compute such term in each processor, that the sum of the two contributions for two adjacent processors gives the exact *rhs* for the shared points. Such method avoids a communication to build the full three-points stencils for the *shared interface* on both cores. The same holds for the treatment of boundary conditions. At the *shared interface*, the boundary conditions are considered only by one processor of the two adjacent ones, whereas on the other core, homogeneous boundary conditions are enforced such that the sum of the two contributions gives the correct boundary value. Thus, considering the IB framework, the IB boundary condition in the *rhs* is assembled only in the processor where both **point_f** and **point_n** are available. In the adjacent processor the *rhs* is assumed to be null so that, once the problem is solved in that direction, the sum of the two *rhs* values for the shared point gives exactly the values of the boundary velocity $\mathbf{U}_b$.

The next step is to reconstruct the new Schur complement that is, in general, influenced by these modifications. Thanks to the fact that the data about the immersed surface are shared among all the processors, each one is able to define which grid line is cut by the IB and which one is not concerned with the presence of a body. Indeed, each processor defines a bounding box for the immersed surface, computing, for each direction, the lowest and the highest index for the grid points that enclose the IB. Inside such regions the initialized matrix will be modified and so the Schur complement. On the other side, for the grid lines that are outside of the bounding box, no matrix will be influenced by the immersed boundary implementation and there is no need to recompute the Schur complement which remains unchanged respect to its initialization. The assembly of the complement requires to communicate each processor's contribution needed in the complete Schur system for each velocity component and for each grid line which is affected by the presence of the IB. To speed up this step, an overlap between the computation and the communication is implemented thanks to the MPI_IALLGATHER routine, such that each processor communicates its own contribution to the Schur complement for all the systems linked with a velocity component while the computation starts to compute the processors' contribution for the other velocity components.

5. Force computation

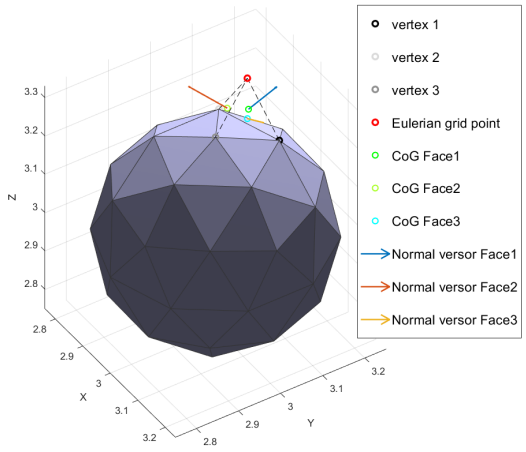
In this section, the strategy adopted for the computation of the forces exerted by the flow on the immersed object is described. Considering the potentiality of the IBM method in simulating complex flows with flexible objects, the approach for the force computation has to be able to compute the local stress on the immersed surface. Several approaches are available to compute the fluid stress tensor σ and to integrate it on the immersed surface to get the force exerted on the immersed boundary:

$$\mathbf{F}_{IB} = \int_{\partial\Omega_{IB}} \sigma \cdot \mathbf{n}_{out}\, dS\,, \tag{32}$$

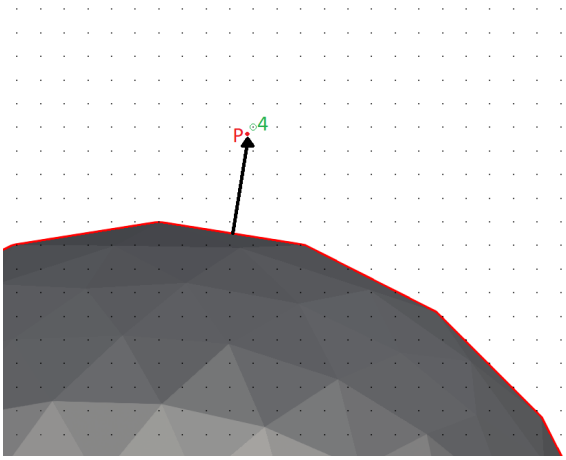
where $\mathbf{n}_{out}$ is the normal versor on the IB surface pointing outwards and $\sigma = -pI + \frac{1}{Re}(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)$, where I is the identity matrix. A possible approach is to directly compute the fluid stress tensor and integrate it over the IB surface $\partial\Omega_{IB}$. This step requires the interpolation of the pressure field on the immersed surface and the computation of the derivatives of the velocity field. This is done defining a Lagrangian marker on the IB mesh and additional points, for each direction, to be able to compute each term of the tensor $(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)$ with a finite difference scheme. However, in general, the computation of the derivative terms enhances the presence of disturbances which can lead to a force field on the IB mesh not smooth. This situation is clearly not desirable for simulating flexible objects. The proposed approach is based on the integral form of the momentum equation of the incompressible Navier-Stokes system and it is built to avoid the direct computation of the derivatives and interpolation techniques to pass data from the Eulerian grid to the Lagrangian IB mesh. Let consider the integral form of the momentum equation:

$$\int_\Omega \frac{\partial\mathbf{u}}{\partial t}\, dV + \int_{\partial\Omega} \mathbf{u}(\mathbf{u}\cdot\mathbf{n})\, dS = \int_{\partial\Omega} \sigma\cdot\mathbf{n}\, dS + \int_\Omega \mathbf{f}\, dV\,, \tag{33}$$

where Ω is the fixed, simply connected control volume considered. This strategy can be used to compute the global force contribution in (32) considering a control volume which encloses the IB. However, here, the control volume used to compute the local stress on the immersed surface is built in a different way. In particular, let us consider the situation of Figure 7(a), an Eulerian grid point is assigned to each triangle of the body mesh to build a tetrahedral control volume. The construction of such tetrahedral element is shown in Figure 7(b). In particular, an auxiliary point P is defined starting from the center of gravity of the three vertices and moving on the normal versor pointing outwards. The distance between the point P and the starting point is equivalent to the height of an hypothetical equilateral tetrahedron with the edge size measuring the average value of the three edges of the triangle of the IB mesh. This step allows one to build a tetrahedral element which is quite smooth and not distorted. However, the auxiliary point P is not in general an Eulerian grid point and, to avoid interpolations, the fourth point of the tetrahedron is chosen as the computational point of the Eulerian grid closest to P.



(a) Tetrahedral element used for the local force computation



(b) Auxiliary point P and Eulerian grid point (4) used to build the tetrahedral Element

Figure 7: Tetrahedral element

Once the control volume has been defined, the integrals in (34) must be discretized with an appropriate strategy.

First of all, it is possible to decompose the surface integral, as: $\int_{\partial\Omega} = \int_{\partial\Omega_1} + \int_{\partial\Omega_2} + \int_{\partial\Omega_3} + \int_{\partial\Omega_\triangle}$, where $\partial\Omega_\triangle$ is the face on the IB mesh, whereas the other three contributions are associated with the other three faces of the tetrahedron. It is possible to rewrite (34) considering this subdivision of the surfaces to obtain:

$$\int_\Omega \frac{\partial \mathbf{u}}{\partial t}\, dV + \sum_{i=1}^{3} \int_{\partial\Omega_i} \mathbf{u}(\mathbf{u}\cdot\mathbf{n})\, dS - \sum_{i=1}^{3} \int_{\partial\Omega_i} \sigma\cdot\mathbf{n}\, dS - \int_\Omega \mathbf{f}\, dV = \int_{\partial\Omega_\triangle} \sigma\cdot\mathbf{n}\, dS - \int_{\partial\Omega_\triangle} \mathbf{u}(\mathbf{u}\cdot\mathbf{n})\, dS = -\mathbf{F}_{IB}\,. \tag{34}$$

The force exerted by the flow on the IB surface is the opposite of the rhs because the normal versor $\mathbf{n}$ is pointing outwards from the tetrahedral element and therefore entering the IB mesh, on the contrary of (32), where the normal is considered to be outgoing from the immersed surface. Moreover, the additional contribution $\mathbf{u}(\mathbf{u}\cdot\mathbf{n})$ is null when the no penetration boundary condition is ensured on the IB. Each integral is done by the midpoint quadrature technique. Thus each term has to be computed in the center of mass of each face. The recovery of each term in such points is done thanks to the Finite Element Method. Indeed, once the tetrahedral element is defined with the four vertices $\mathbf{v}_i = (x_i, y_i, z_i)$, with $i = 1, 2, 3, 4$, it is possible to compute the FEM shape function N_i, to recover the approximate velocity field in each point of this element. The linear system to solve is:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ y \\ z \end{bmatrix}\,. \tag{35}$$

The velocity field in a generic position is computed as:

$$\mathbf{u}(x,y,z) = N_1(x,y,z)\mathbf{u}_1 + N_2(x,y,z)\mathbf{u}_2 + N_3(x,y,z)\mathbf{u}_3 + N_4(x,y,z)\mathbf{u}_4\,, \tag{36}$$

where $\mathbf{u}_i = \mathbf{u}(x_i, y_i, z_i)$, $i = 1, 2, 3, 4$ is the known velocity on the four vertices. Thanks to the FEM approach, the computation of the derivatives simplifies to:

$$\frac{\partial u_k}{\partial x_j} = \frac{\partial N_1}{\partial x_j} u_{1,k} + \frac{\partial N_2}{\partial x_j} u_{2,k} + \frac{\partial N_3}{\partial x_j} u_{3,k} + \frac{\partial N_4}{\partial x_j} u_{4,k}\,, \tag{37}$$

where $u_{i,k}$ stands for the k-component of the velocity of the i vertex. Considering a four-point, linear tetrahedral element, the derivatives are constant. Particular attention must be paid to avoid the construction of very distorted elements, for which the computation of the derivatives can be significantly inaccurate. Indeed, the strategy presented above, allows one to avoid that the four points of the tetrahedral element are aligned, which can lead to errors in the derivative computation. The correct construction of such element is guaranteed when the Eulerian grid is sufficiently fine with respect to the IB triangulation, otherwise the fourth point could be in a quite random position with respect to the three vertices of the IB triangle, resulting in a loss of accuracy in the computation of the viscous stresses.
Once the velocity and its derivatives are computed, a strategy to derive the pressure field in the tetrahedron has to be defined. The pressure field is known only on the computational grid point and a constant extrapolation is used to assume a constant value for all the tetrahedron. The volume forces $\mathbf{f}$ and the unsteady term can be evaluated on the center of mass of the tetrahedron.

6. Numerical results

In this section, the method is validated through numerical testing and the convergence analysis is reported. The first analysis concerns the simplified version of the full 3D method and preliminary tests are done solving the one-dimensional diffusion equation with both a fixed and a moving IB inside the computational domain. Then, the IB method is tested in its full 3D version to check the convergence results with manufactured solutions and reference cases. The computation of the force which acts on the IB is also tested with an analytical solution. In particular, a steady manufactured solution is used to compute the forces with several IB triangulations and the error with respect to the exact analytical solution is used to check the convergence rate.

A scalability test is done to verify the parallel performance of the implemented code and to compare it with the original code [7].

Once the convergence analysis is completed, the proposed IBM is used to simulate a realistic flow. The reference case used to verify the IBM is the incompressible flow past a sphere, which is a well established benchmark

in the literature. In particular, as the implicit immersed boundary treatment of this method seems to be a good alternative to the explicit one for low Reynolds number flows, simulations are done at varying Re, that ranges from 0.1 to 10, to verify if this approach is able to capture the dependence of the drag coefficient to the Reynolds number. Another test is done by considering the transition of the uniform flow past a sphere, from an axial-symmetrical condition to a planar-symmetrical one for a critical Re $\approx$ 212, verifying the computation of both lift and drag contributions.

6.1. Convergence analysis

6.1.1 1D Heat equation with IB

The IB method is tested preliminarily to solve a one-dimensional heat equation with a boundary that is in a generic position inside the computational domain. The direction-splitting approach introduced in Section 2 allows us to decompose the 3D problem of an object moving in a fluid as a sequence of 1D problems. Here we verify the accuracy of this IB treatment and its possible limits on the time steps.
The problem considered, with a body either moving or not, is:

$$\frac{\partial u(x,t)}{\partial t} - \frac{\partial^2 u(x,t)}{\partial x^2} = f(x,t) \ , \tag{38}$$

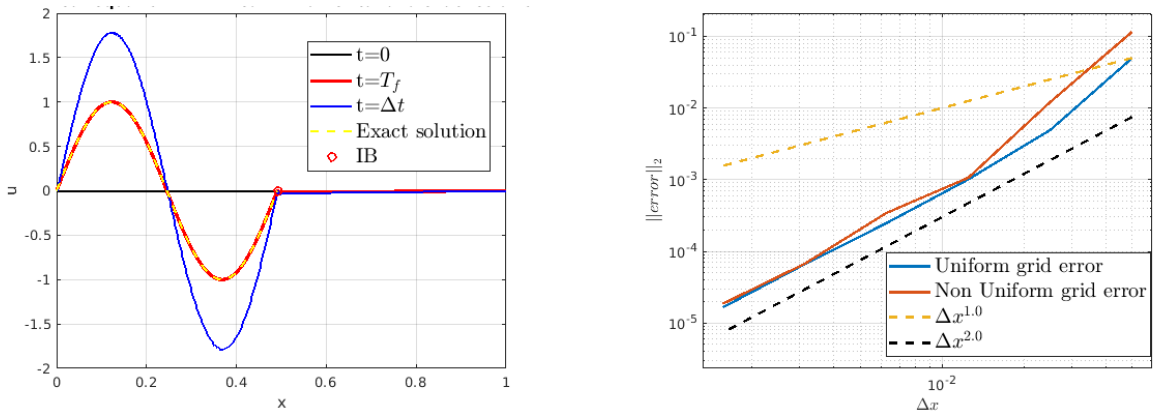
in $x \in [a,b], t \in (t_0, T_{end})$ with initial and boundary conditions:

$$\begin{cases} u(x,t_0) = u_0(x) & x \in [a,b] \ , \\ u(a,t) = u_A(t) & t \in (t_0, T_{end}) \ , \\ u(b,t) = u_B(t) & t \in (t_0, T_{end}) \ . \end{cases} \tag{39}$$

Fixed body First, we test our scheme with a still body in position x_b with a constant Dirichlet boundary condition u(x_b,t)=0. The computational domain is Ω=[0,1] and x_b is such that $0 < x_b < 1$. The fluid domain of interest is the portion $\Omega_f = [0, x_b]$. The manufactured solution for (38) is:

$$u_{exact}(x) = \sin(2\pi \frac{x}{x_b}) \ , \tag{40}$$

that corresponds to the forcing term $f(x,t) = 4\frac{\pi^2}{x_b^2}\sin(2\pi\frac{x}{x_b})$ with null initial and boundary conditions.
The integration is carried out from t_0=0 to T_{end}=100 with a time step Δt=0.1 and a Crank-Nicolson scheme. The grid convergence is tested on the steady state solution, dividing the computational grid Ω in N intervals, with N=[20, 40, 80, 160, 320, 640].
In the Figure 8, the solution and the grid convergence are reported for both a uniform and a quasi-uniform grid. The results computed on the steady solution with $L^2(\Omega_f)$ norm confirm the second order accuracy.



(a) 1D Heat equation with fixed IB: Numerical and exact solution

(b) Spatial convergence of the IB method: uniform and quasi-uniform grid

Figure 8: 1D Heat equation with a fixed IB

Moving IB The second test is performed on a moving body. The reference test used to compare the numerical solution is provided by Morland [25]. The problem to solve is the heat equation (38) without source term $f(x,t) = 0$, defined in time $t \in (0, T_f)$ and in the domain $x \in [0, s(t)]$. The boundary and initial conditions are:

$$\frac{\partial u}{\partial x} = -g(t), \quad x = 0, \ t > 0, \tag{41a}$$

$$u = p(x), \quad \frac{\partial u}{\partial x} = -\frac{ds}{dt}, \quad x = s(t), \ t > 0, \tag{41b}$$

$$u = 0, \quad t = 0, \ 0 < x < s(0), \tag{41c}$$

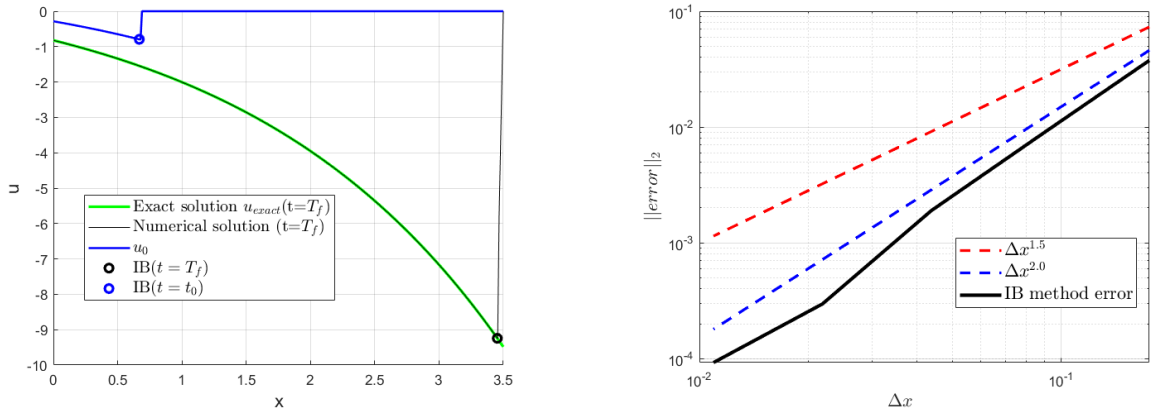
with $g(t) = \frac{1}{2}\exp(\frac{1}{4}t)$, $p(s) = 2[1 - \exp(\frac{1}{2}s)]$, and $s(0) = 0$.
The analytical solution to this problem is:

$$u(x,t) = 1 - \exp(\frac{1}{2}x + \frac{1}{4}t), \quad 0 \le t < 4\ln 2, \tag{42a}$$

$$s(t) = -2\ln[2 - \exp(\frac{1}{4}t)], \tag{42b}$$

$$\dot{s} = \exp(\frac{1}{2}s) - 0.5. \tag{42c}$$

We consider a one-dimensional domain $\Omega = [a, b]$, with $a = 0$ and $b = 3.5$. The domain is divided in N uniform intervals, with N = [20, 40, 80, 160, 320, 640]. The simulation runs from $t \in (t_0, T_f)$ with $t_0 = 1.0$ and $T_f = 2.4$ with 2000 time steps and with a Crank-Nicolson scheme. The time step is selected to keep the error due to temporal discretization sufficiently low but also to avoid that the boundary crosses more than one interval in one step. The numerical solution is initialized with the exact solution in $0 < x < s(t_0)$ and zero elsewhere. The boundary moves with a speed equal to $\dot{s}$.
The results are shown in Figure 9. The numerical solutions at time t_0 and T_f are shown with their respective value and position of the moving IB. The convergence analysis is reported in Figure 9 where the error of the numerical solution with respect to the analytical one in the L^2 norm is plotted against the interval size. The error is computed only on Ω_f at the time step $t = T_f$. The plot testifies the second order spatial accuracy of this method also in the presence of a moving IB.



(a) Morland problem : analytical and numerical solution.

(b) Morland problem : spatial convergence.

Figure 9: 1D Heat equation with a moving IB : Numerical solution and convergence analysis

6.1.2 3D diffusion equation

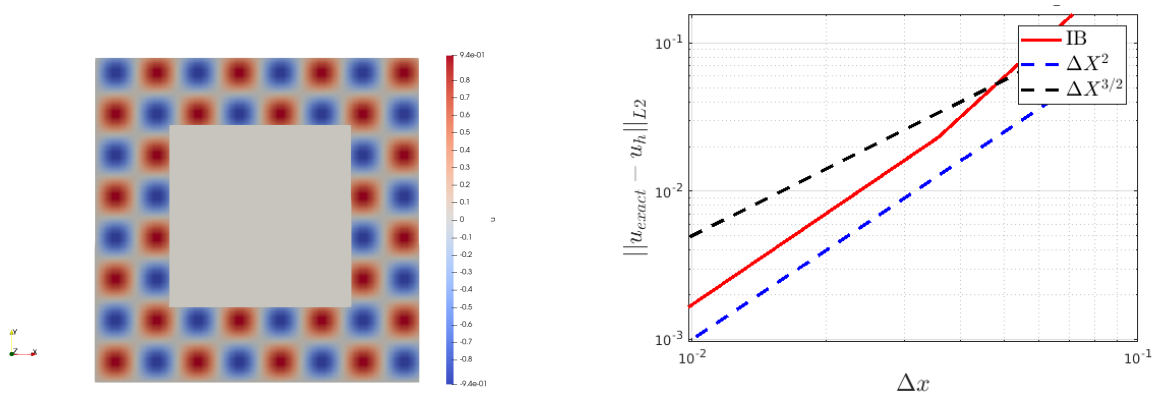
The next test is performed solving the three-dimensional diffusion equation. To test the convergence of the IB, a manufactured solution is built and an IB is included into the computational domain to verify that it does not alter the expected second order spatial accuracy.
The equation to be solved is:

$$\frac{\partial u}{\partial t} - \nabla^2 u = f \ , \tag{43}$$

defined in a cubic domain $\Omega = [0,2] \times [0,2] \times [0,2]$ and for $t \ge 0$.

Spatial convergence The spatial convergence is tested with a fixed time step and a uniform grid with different stencil sizes. A manufactured solution is defined and the source term f is computed so as to produce the solution. The solution is used to define boundary conditions on both $\partial\Omega$ and the IB surface. The manufactured solution is: $u_{exact} = \sin(4\pi x)\sin(4\pi y)\sin(4\pi z)$ that is obtained imposing a source term $f = 48\pi^2 u_{exact}$. The initial condition is $u(x,0) = 0$ and boundary conditions are $u|_{\partial\Omega} = u|_{\partial Body} = 0$. The immersed boundary is a cube with coordinates $[0.5, 1.5] \times [0.5, 1.5] \times [0.5, 1.5]$.
The computational domain is divided uniformly with a number of intervals for each direction equal to N = [14, 28, 56, 102]. The time step is $\Delta t = 10^{-3}$ to keep the temporal discretization error low enough and the number of time steps is 1000, necessary to reach a steady solution with a Reynolds number equal to 1. The convergence is analyzed comparing the steady numerical solution with the exact one. The simulation is performed with the Crank-Nicolson scheme.

The numerical results are shown in Figure 10. A slice in a plane $z = 1.125$ shows the steady solution and the immersed cube. The convergence plot proves the second-order spatial accuracy of the method in the $L^2(\Omega_f)$ norm. The convergence analysis is done by plotting the error against the stencil size $\Delta x = \frac{2}{N}$.



(a) Slice on the plane z = 1.125. The grey zone is the slice of the cubic IB.

(b) Manufactured solution: convergence analysis.

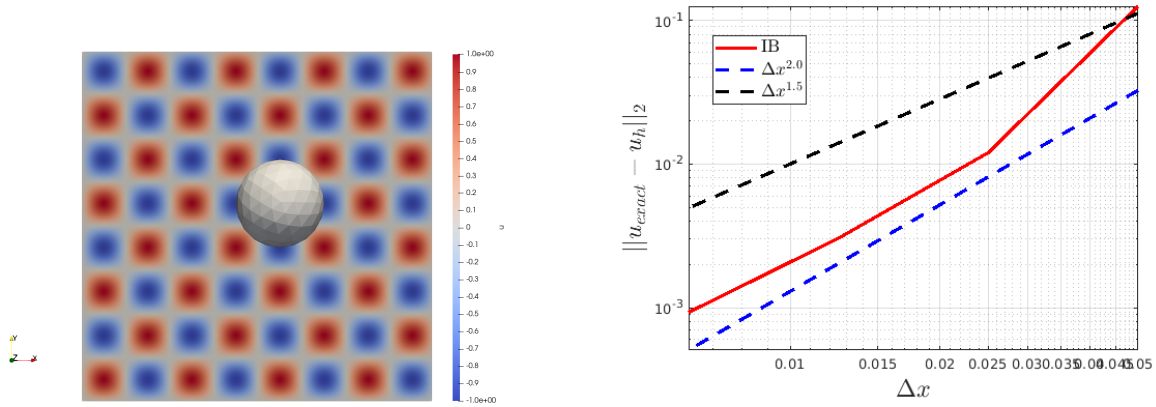
Figure 10: 3D Heat equation with a stationary IB : Numerical solution on a plane (left) and convergence rate (right)

Temporal and spatial convergence In these simulations both the spatial and the temporal discretization is refined keeping a constant ratio between the interval grid size and the time step. In this way, it is possible to check if the IB treatment introduces an additional temporal error that destroys the desired second order convergence rate. The tests are performed with the Crank-Nicolson method. The setup of the simulations is defined by a cubic computational domain $\Omega = [0, 2] \times [0, 2] \times [0, 2]$ and a time interval of $t \in (0, 1)$. The computational grid is uniform. The discretization corresponds to a number of intervals equal to $N_h = [20, 40, 80, 160]$ and a number of time steps equal to $N_t = [25, 50, 100, 200]$.

The manufactured solution for the problem (43) is : $u_{exact}(x,t) = \sin(\frac{5}{2}\pi t)\sin(4\pi x)\sin(4\pi y)\sin(4\pi z)$ that is obtained with the source term : $f(x,t) = (48\pi^2 \sin(\frac{5}{2}\pi t) + \frac{5}{2}\pi\cos(\frac{5}{2}\pi t))\sin(4\pi x)\sin(4\pi y)\sin(4\pi z)$. The IB treatment is tested with the insertion of the body in a generic position inside Ω. In this case, a sphere of radius $R = 0.25$ centred in the point $C = (1.12, 1.12, 1.12)$. The exact solution is used to define both the initial and the boundary conditions on $\partial\Omega_f$ and on the IB surface. The convergence analysis is performed at the final time $t = 1.0$. In Figure 11, the slice of the solution with the plane $z = 1.12$ is shown. The error is computed with the $L^2(\Omega)$ norm only in the fluid domain. The convergence plot in Figure 11 shows the expected second order accuracy.

6.1.3 Navier-Stokes equations

To conclude the convergence analysis, the Navier-Stokes equations are considered and an IB is introduced in a generic position inside the domain. The computational domain Ω is a cube : $\Omega = [0, 6] \times [0, 6] \times [0, 6]$. The simulations run in the time interval $0 < t \leq T = 1$. The manufactured solution of the problem is:



(a) Slice on the plane z = 1.12. The spherical IB is represented with its surface triangulation

(b) Convergence analysis with spherical IB at constant ratio between Δt and Δx

Figure 11: 3D Heat equation with a spherical IB : Numerical solution on a plane and convergence results

$$u = \sin(x)\,\cos(t+y)\,\sin(z)\,, \tag{44a}$$
$$v = \cos(x)\,\sin(t+y)\,\sin(z)\,, \tag{44b}$$
$$w = 2\cos(x)\cos(t+y)\cos(z)\,, \tag{44c}$$
$$p = \frac{3}{Re}\cos(x)\cos(t+y)\cos(z)\,. \tag{44d}$$

The velocity field is divergence-free to satisfy the incompressibility constraint. The exact velocity field is used to define the initial condition and is adopted as Dirichlet boundary condition on $\partial\Omega$ and on the immersed boundary surface.

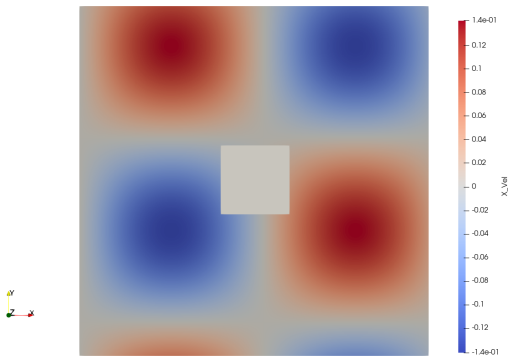
The force field is obtained substituting the manufactured solution into the Navier-Stokes equations:

$$\begin{aligned}
f_x &= \frac{\sin(x)}{Re}[\frac{1}{2}(Re\cos(x)(1\ +\ 2\ \cos(2(t+y)) + \cos(2z)))+ \\
&\quad - 3\cos(t+y)(\cos(z) - \sin(z)) - Re\sin(t+y)\sin(z)]\,, \\
f_y &= \frac{1}{4Re}[Re\cos^2(x)(3+\cos(2z))\sin(2(t+y)) - 2Re\sin^2(x)\sin(2(t+y))\sin^2(z)+ \\
&\quad + 4\cos(x)(-3\cos(z)\sin(t+y) + (Re\cos(t+y) + 3\sin(t+y))\sin(z))]\,, \\
f_z &= \frac{1}{Re}[\cos(x)(-2Re\cos(z)\sin(t+y) + \cos(t+y)(6\cos(z) - 3\sin(z)))+ \\
&\quad - \frac{1}{2}(Re\cos^2(x)(3+\cos(2(t+y)))sin(2z)) - Re\cos^2(t+y)\sin^2(x)\sin(2z)]\,.
\end{aligned} \tag{45}$$

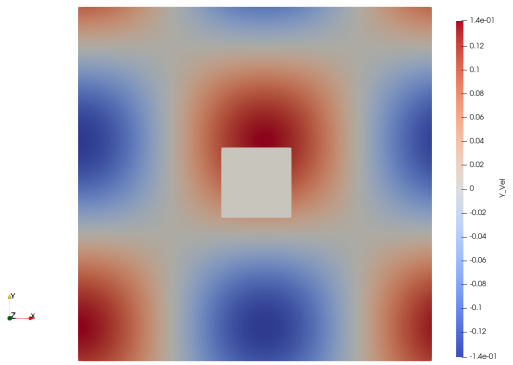
The convergence is verified on the solution at time T. The problem is solved with $Re = 1$ to avoid fluid-dynamic instabilities. In the next paragraphs the spatial and temporal convergence of the IB method is tested.

Spatial convergence The spatial accuracy is tested on uniform grids discretized by N points for each direction, with N = [10, 20, 40, 80, 160]. The time step is set as $\Delta t = 0.003125$. A cubic IB is introduced inside the computational domain Ω in a generic position that does not conform with the grid. In particular, the IB has the following coordinates: $\Omega_{IB} = [2.43, 3.56] \times [2.43, 3.56] \times [2.43, 3.56]$. The fluid domain is the portion $\Omega_f = \Omega \backslash \Omega_{IB}$. The Crank-Nicolson scheme is adopted for these simulations and the solution at the final time is shown in Figure 12 and 13. The convergence analysis points out the second order accuracy in space of the IB method for both the velocity and the pressure field, until the temporal error is negligible. The plot in Figure 14 compares the $L^2(\Omega_f)$ norm of the error of the numerical solution with respect to the manufactured one against the grid interval size Δx.

Constant CFL convergence The successive test is done refining at the same time the grid size and the time step. The simulation is done with a constant CFL to point out the convergence rates of the IB method. In

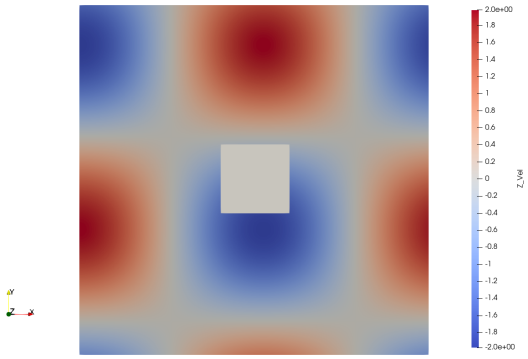


(a) X-component of the velocity: Slice on the plane z = 3.0. The cubic IB is shown in grey.

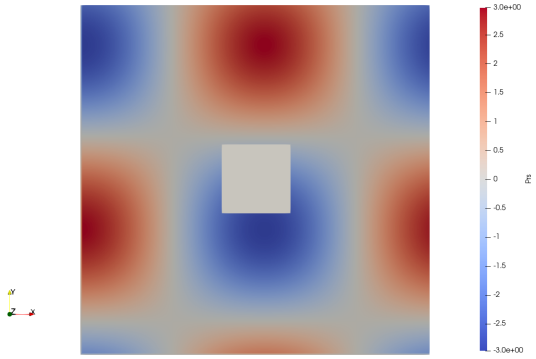


(b) Y-component of the velocity: Slice on the plane z = 3.0. The cubic IB is shown in grey.

Figure 12: 3D Navier-Stokes manufactured solution with a cubic IB : Numerical solution on a plane (1)

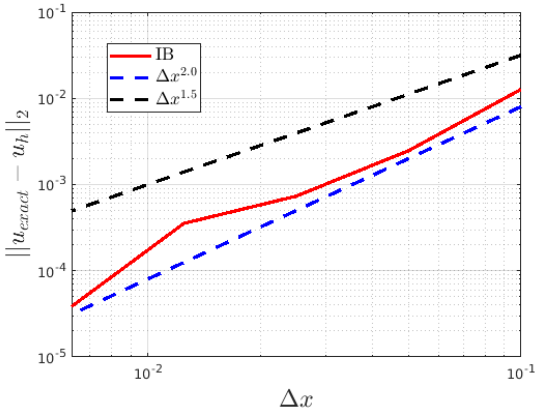


(a) Z-component of the velocity: Slice on the plane z = 3.0. The cubic IB is shown in grey.

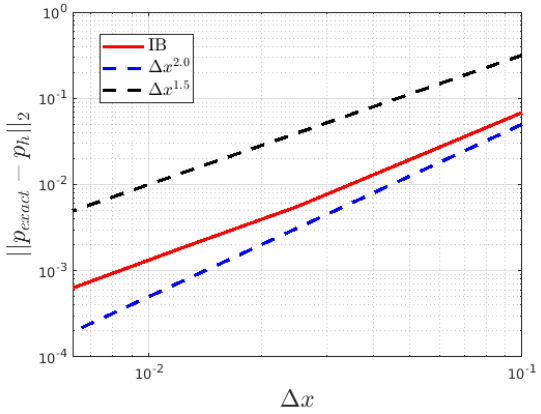


(b) Pressure: Slice on the plane z = 3.0. The cubic IB is shown in grey.

Figure 13: 3D Navier-Stokes manufactured solution with a cubic IB : Numerical solution on a plane (2)



(a) Velocity: spatial convergence



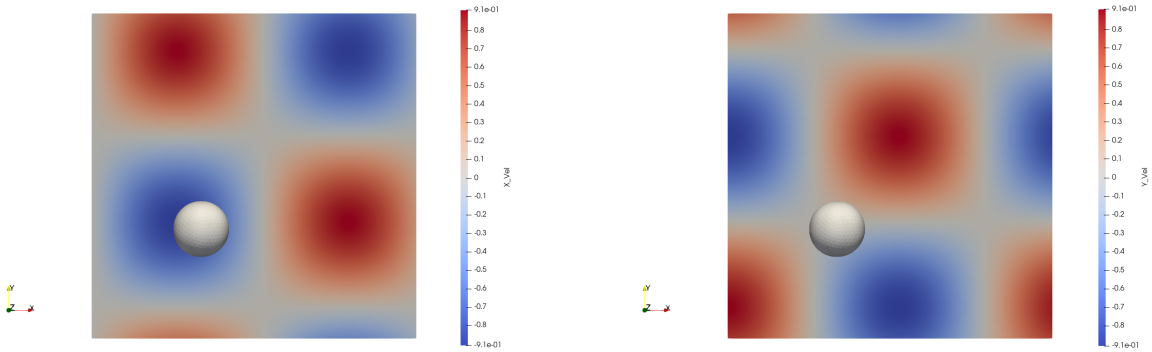
(b) Pressure: spatial convergence

Figure 14: 3D Navier-Stokes manufactured solution with a cubic IB : spatial convergence plots

particular, the refinement of both the stencil size and the time integration interval allows one to reduce at the same time the temporal and the spatial error. This procedure extends the range where the second order rate is noticeable.

The manufactured problem adopted and the computational domain are the same as in the previous case.

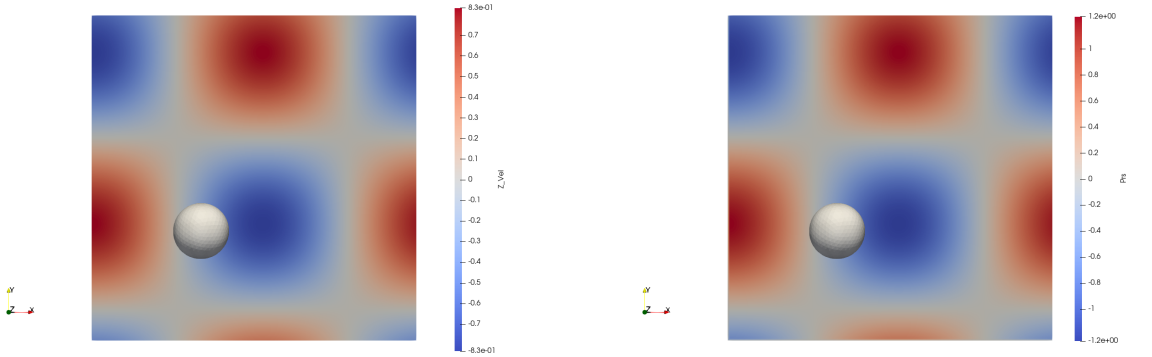
The grid is uniformly discretized with a number of points for each direction defined by N = [10, 20, 40, 80, 160] whereas the time step is $\Delta t = 0.1 \times 2^{-n}$, with n that ranges from 0 to 4. This discretization ensures a constant CFL = 1/3. The immersed body is a sphere centred in the point $C = (2.0, 2.0, 2.0)$ with a radius $R = 0.51$. The numerical solution in the plane $z = 2.0$ at the final time step $t = 1$ is shown in Figure 15 and 16, including also the immersed sphere. The convergence analysis is done computing the L^2 norm of the error in the fluid domain of the numerical solution at the final time step with respect to the manufactured solution (44). The convergence rates, available in Figure 17, confirm the expected second order accuracy for the velocity while the convergence rate for the pressure field is between 1.5 and 2.



(a) X-component of the velocity: Slice on the plane z = 2.0. The spherical IB is shown in grey.

(b) Y-component of the velocity: Slice on the plane z = 2.0. The spherical IB is shown in grey.

Figure 15: 3D Navier-Stokes manufactured solution with a spherical IB : Numerical solution on a plane (1)



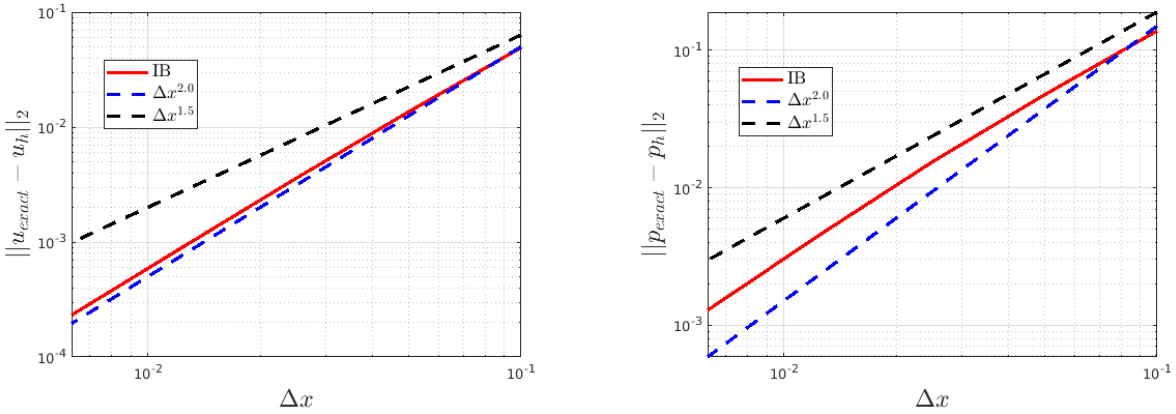
(a) Z-component of the velocity: Slice on the plane z = 2.0. The spherical IB is shown in grey.

(b) Pressure: Slice on the plane z = 2.0. The spherical IB is shown in grey.

Figure 16: 3D Navier-Stokes manufactured solution with a spherical IB : Numerical solution on a plane (2)

6.2. Test of force calculation

In this section, the approach used to compute the forces acting on a IB surface, described in Section 5, is tested. The manufactured problem used above for the Navier-Stokes convergence is used. However, the time dependency is dropped and the steady case at $t = 0$ is considered. The forcing terms are computed to satisfy the incompressible Navier-Stokes model with such analytical solution. The scheme is tested in the same computational domain $\Omega = [0, 6] \times [0, 6] \times [0, 6]$ discretized uniformly in the three directions with an interval size equal to 0.00625 in the interval [2.75, 3.25] and increased outwards with an expansion rate of 5% outwards. The number of grid points is: $208 \times 208 \times 208$. The IB is a sphere centred in $C = (3.0, 3.0, 3.0)$ with radius $R = 0.25$. The scheme is tested at Re = [1.0, 100]. In Figure 18, the relative error of the force is plotted against the non-dimensional ratio $\frac{h}{D}$ between the average size of the triangle edge of the IB mesh and the diameter of the sphere. Both images testify the convergence of the method. In particular, it is worth noticing that in the case of $Re = 1$, the first order error dominates for all the grids since the viscous contribution is the dominant



(a) Velocity: convergence rates at constant CFL

(b) Pressure: convergence rates at constant CFL

Figure 17: 3D Navier-Stokes manufactured solution with a cubic IB : Convergence at constant CFL.

one. At $Re = 100$, the first order error becomes the dominant contribution only for the finest grids, since the viscous contribution is no longer the most important one and the first order accuracy appears only when the second order contribution for the pressure and the momentum flux has become sufficiently small.

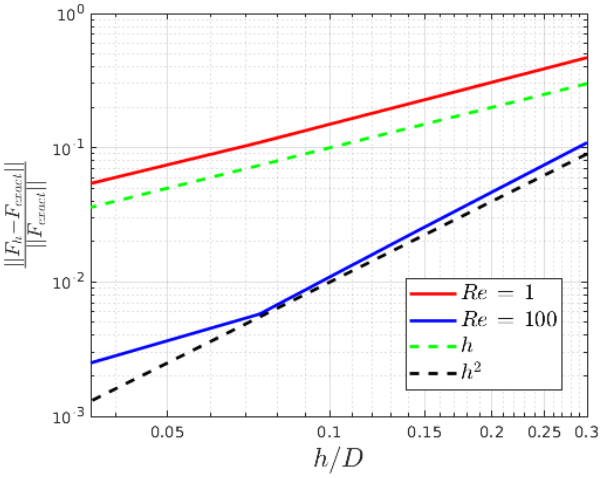


Figure 18: Force convergence with a steady manufactured solution

6.3. Scalability test

The scalability performance is tested with GALILEO100 supercomputer at CINECA. This is a cluster equipped with 636 computing nodes, each with two *Intel CascadeLake 8260* with 24 cores each, 2.4 GHz, 384GB RAM and Mellanox Infiniband interconnection (100 Gb/s). The adopted problem is the same used before for the Navier-Stokes convergence analysis. The strong scalability analysis is carried out keeping the size of the problem fixed. Indeed, 400^3 computational points are considered, distributed among a number of processors in the range $N_p = [16, 32, 64, 128, 256, 512, 1024]$.
The results are compared with the original code [7] which does not include the possibility of treating an IB. For such reason, the the original code is tested without the introduction of an immersed object, whereas the implemented code uses an immersed sphere centred in $C = (2.0, 2.0, 2.0)$ with radius $R = 0.5$.
The results are shown in Figure 19 where the CPU time per time step and grid point is shown at varying number of processor N_p. The ideal behaviour is a constant value. The original version scales better, with a slight loss of performances for few cores recovered by a more than ideal behaviour for $N_p > 128$. The new code with the IB treatment has a reduced efficiency with a growing number of processors, with a CPU time that doubles from 16 to 1024 cores. The new code is slower with respect to the original code [7] and it has a CPU time per time step and grid point in the order of 10^{-6}. There are several aspects which affect the non

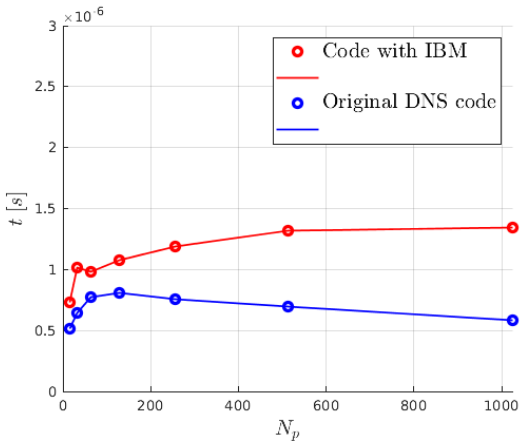


Figure 19: Strong scalability test: comparison with the original code. Number of processors VS CPU time per time step and grid point

ideal behaviour and the differences with respect to the original version. It must be considered that the original code uses the factored version of the Thomas algorithm with respect to the new code causing an increase in the floating point operations in the back-substitution phase, with a loss of serial performance. Another point is the different numerical method adopted. Indeed, the modified scheme (14) described in Section 2.1 is less efficient with respect to the original scheme, requiring the update of the *rhs* for every directional step in the solving phase. Moreover, the reconstruction of the intermediate variables for the interface values requires an additional communication which is not required in the original scheme. In particular, the velocity field, obtained as the solution of the final step, has the correct values for the interfaces unknowns in all the three directions, while the intermediate, temporary variables η and ζ do not satisfy such property because of the parallel treatment of the *rhs*. Indeed, since they are not the result of all three directional problem, the interface values must be summed among adjacent processors. The other important aspect is the reconstruction of the Schur complement. Indeed, the original DNS code builds the Schur matrix once and for all in the pre-processing phase but the new code requires it for each time step. To speed up this routine, the overlap between computation and communication is implemented as explained in Section 4.3.

6.4. Flow past a sphere

The immersed boundary method, which is previously verified to be second order accurate for the velocity field with a manufactured solution, is now tested on a realistic case. Hence, in this section, an incompressible flow past a sphere is considered. The previous tests in Section 6.1, verify that the introduction of an IB does not alter the accuracy of the method and the immersed surface is transparent with respect to the velocity and pressure field of the manufactured solution. However, in a realistic case, no forcing term guarantees that the analytical solution is obtained in all the computational domain, also inside the IB. In the simulation of a realistic flow, both the velocity and the pressure fields are only continuous across the IB and the points inside the solid body have no meaningful values. This aspect has to be further investigated with realistic flows simulations, to verify that the accuracy of the method is preserved also in this case. The incompressible flow past a rigid fixed sphere is chosen as a test case because of its available, well established literature. Such flow is numerically simulated to check if the proposed method is able to correctly compute the velocity and the pressure field and the forces acting on the body.

6.4.1 Drag coefficient VS Reynolds number

The Reynolds number defines the flow state for the incompressible flow around a sphere. Here, the method is tested for three different values $Re = [0.1, 1.0, 10]$. In particular, the range of Re is chosen to verify the accuracy of the method at very low Reynolds number, for which the implicit treatment of the IB guarantees significant benefits with respect to the explicit correction of the velocity field. In such regime, the flow is expected to be axis-symmetric [17] and there are no separations in the wake of the body. Indeed, the early transitions occur at higher Reynolds number [6]. In particular, the formation of the separation region, with a detached flow behind the sphere, starts at higher Reynolds number, approximately at $Re \approx 24$ [37], whereas the axis-symmetric structure of the wake breaks up for $Re = 212$. Then, the wake remains steady, non axis-symmetrical, until a transition from the steady regime to the unsteady periodic one occurs at a critical Reynolds number $Re = 273$

[6].
In the range of Reynolds number adopted, several numerical or experimental results are available in the literature. In general, the dependence of the drag coefficient, C_d, with respect to Re can be expressed, for very low Re, by the Stokes's solution. Stokes provides the expression of the drag of a sphere immersed in a uniform flow: $Drag = 6\pi\mu UR$, where R is the radius of the sphere, U the freestream velocity and μ the fluid viscosity. This condition is valid for the *creeping flow* around a sphere, and it was obtained by an analytical solution of the incompressible Navier-Stokes equations, assuming a steady axis-symmetric flow and neglecting the convective term [21]. Such assumption is valid only when Re is sufficiently low to guarantee that such an approximation is accurate. The drag coefficient for the *creeping flow* around a sphere becomes:

$$C_d(Re) = \frac{24}{Re} \, . \tag{46}$$

The Stokes' solution implies symmetrical streamlines behind and in front of the sphere, hence the wake is absent. This phenomenon comes from the elimination of vorticity convection once the inertia terms are neglected [21]. In the range of Reynolds number of the order of one, Oseen provides a satisfactory correction to the Stokes' prediction [6] to consider these effects, the drag coefficient becomes:

$$C_d(Re) \, = \, \frac{24}{Re}\left(1 + \frac{3}{16}Re\right) \, . \tag{47}$$

For higher Reynolds numbers, however, the functions which express the drag coefficient as a function of the Reynolds number comes from a fit of experimental results. Morrison [26] provides a data correlation curve which fits the experimental data for Re up to 10^6:

$$C_d(Re) \, = \, \frac{24}{Re} + \frac{2.6\left(\frac{Re}{5.0}\right)}{1+\left(\frac{Re}{5.0}\right)^{1.52}} + \frac{0.411\left(\frac{Re}{263000}\right)^{-7.94}}{1+\left(\frac{Re}{263000}\right)^{-8.00}} + \frac{0.25\left(\frac{Re}{10^6}\right)}{1+\left(\frac{Re}{10^6}\right)} \, . \tag{48}$$

The IBM is tested to verify its accuracy in the computation of the drag coefficient for such flow compared with the available references. The computational domain adopted is a cube $\Omega = [-L,\, L] \times [-L,\, L] \times [-L,\, L]$, with $L = 10$. Different boundary conditions are used for $\partial\Omega$. An inlet (assigned velocity and homogeneous Neumann boundary condition for the pressure) is considered in the plane $x = -L$ with velocity $\mathbf{u}_\infty = [1.0,\, 0.0,\, 0.0]$, whereas an outlet (Dirichlet b.c. for the pressure and homogeneous Neumann b.c. for the velocity) is chosen for $x = L$. Symmetry boundary conditions are adopted for the other faces of the computational domain. The immersed sphere is centred in the point $C = (0.0,\, 0.0,\, 0.0)$ and it has a radius $R = 0.5$, discretized by 1280 triangular faces. A *non-slip wall* boundary condition is assigned for the immersed surface to satisfy the condition of null immersed boundary velocity, $\mathbf{U}_b = 0$. The computational domain is discretized adopting three different grids, which are equally discretized for the three directions x, y and z. All the grids have a uniform interval size region for $[-2R,\, 2R]$, then the interval size grows with a constant expansion ratio. In Table 3, the number of grid points, the smallest interval in the region where the sphere is located and the expansion ratio are shown:

	Number of points	Smallest interval size	Expansion ratio
Grid 1	$150 \times 150 \times 150$	0.04	1.05
Grid 2	$210 \times 210 \times 210$	0.025	1.045
Grid 3	$320 \times 320 \times 320$	0.0125	1.045

Table 3: Set of grids adopted to simulate the incompressible flow past a sphere

The drag coefficient is chosen as the parameter to check the accuracy of the IBM. It is obtained by the dimensionless x-component of the measured force scaled with the frontal area of the sphere: $A_{ref} = \pi R^2$. Two approaches are used to compute such force. The numerical results are provided for both the method described in Section 5, renamed Local approach, and a global approach with a control volume which embeds the body. Such control volume is equivalent to the computational domain and it allows one to compute the global force acting on the sphere.
The numerical simulations correctly predict the flow field described by Stokes for very low Reynolds number. For $Re = 0.1$, the x-component of the velocity in the symmetry plane $z = 0$ in Figure 20 confirms the *upstream-downstream* symmetry. This aspect is also confirmed by the graphs of the pressure and the x-component of the velocity on the line $(-3.0,\, 0.0,\, 0.0) \times (3.0,\, 0.0,\, 0.0)$, see Figure 21 and 22. In these graphs, the region $-0.5 < x < 0.5$ is not considered since it is inside the sphere, thus it does not belong to the fluid domain of interest. Considering $Re = 0.1$, the Stokes model is reasonable, and the velocity is almost front-rear symmetrical

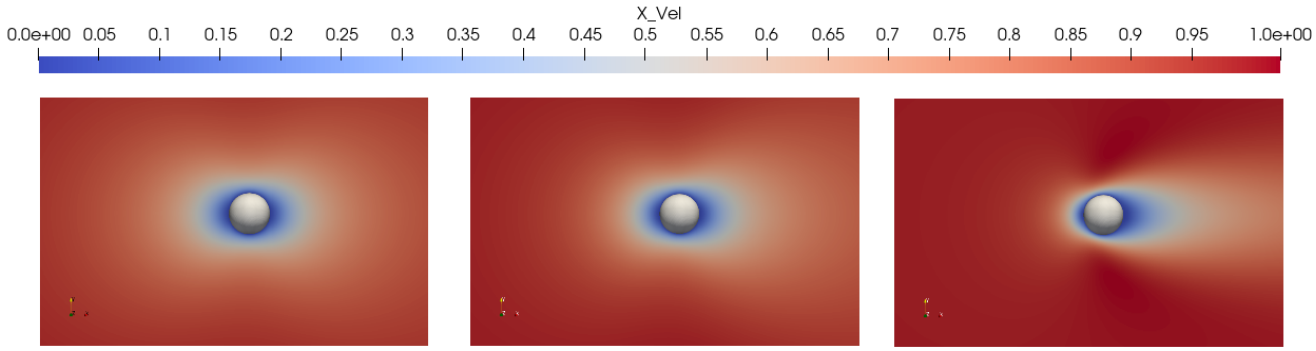


Figure 20: X-component of the velocity in the symmetry plane $z = 0$. *Left*: $Re = 0.1$. *Center*: $Re = 1.0$. *Right*: $Re = 10$.

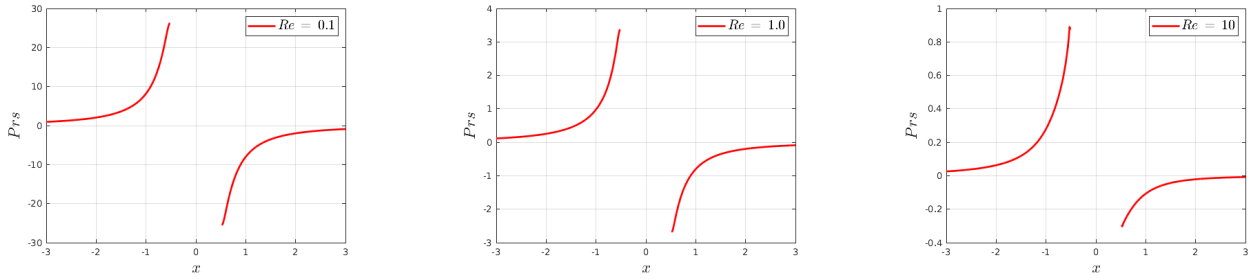


Figure 21: Pressure distribution in front of and behind the sphere. Plot over line: from $(-3.0, 0.0, 0.0)$ to $(3.0, 0.0, 0.0)$ *Left*: $Re = 0.1$. *Center*: $Re = 1.0$. *Right*: $Re = 10$.

(Figure 22). These considerations are no longer valid for a Reynolds number of order one and above, indeed the pressure distribution (Figure 21) and the velocity field (Figure 22) are no longer *upstream-downstream* symmetrical. A wake behind the sphere is present and it becomes more and more extended as Re increases. However, the flow is always axis-symmetrical, as noticeable in Figure 20.
Considering the x-component of the velocity behind the sphere, which has always a not negative value, in Figure 22, it is possible to note that the recirculation region is not present in the wake of the sphere in such regime, which is coherent with the experimental results of [37].
The numerical results provide the drag coefficient in the set of different grids, summarized in Table 4 and 5. These tables collect the results of the same simulation, considering or not the usage of the boundary fitted stencils described in Section 3.3. In particular, it is possible to point out that the introduction of the corrected version of the stencils improves the accuracy of the method because they avoid to use solid point into the *three-point* stencils in the point closest to the interface. In particular, the local approach used to compute the forces is very influenced by the adoption or not of boundary fitted stencil due to the fact that it only uses the first few points around the sphere which are the most affected by the incorrect treatment of the *fluid-solid* discontinuity. The results with boundary fitted stencils are then shown, in Figure 23, to point out the trend of the $C_d(Re)$ curve in comparison with the data correlation fit (48). The numerical results for the drag coefficient fit the curve for the assigned Reynolds number values and confirm the accuracy of the method in the simulation of realistic incompressible flows.

C_d	Re 0.1	Re 1	Re 10
Grid 1	245	26.9	4.31
Grid 2	249	27.1	4.30
Grid 3	252	27.3	4.30

C_d	Re 0.1	Re 1	Re 10
Grid 1	338	32.9	4.61
Grid 2	292	29.8	4.35
Grid 3	292	28.6	4.33

Table 4: Drag coefficient at varying Reynolds number (boundary fitted stencils NOT adopted). Global force approach (left) and local force approach (right)

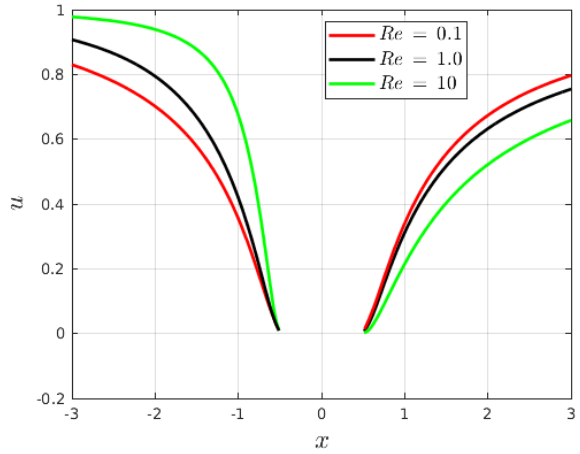


Figure 22: X-component of the velocity. Plot over line: from $(-3.0, 0.0, 0.0)$ to $(3.0, 0.0, 0.0)$ for three different Reynolds numbers.

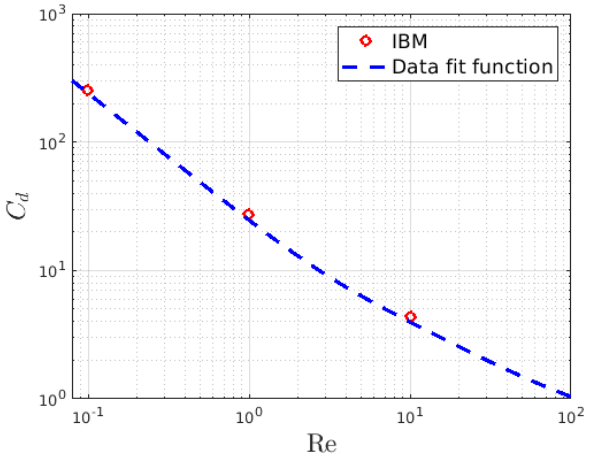


Figure 23: Drag coefficient: numerical results compared with the data fit function by [26]

C_d	Re 0.1	Re 1	Re 10
Grid 1	257	26.0	4.03
Grid 2	256	27.6	4.33
Grid 3	256	27.6	4.33

C_d	Re 0.1	Re 1	Re 10
Grid 1	277	29.2	4.40
Grid 2	257	27.3	4.17
Grid 3	256	27.2	4.14

Table 5: Drag coefficient at varying Reynolds number (boundary fitted stencils adopted). Global force approach (left) and local force approach (right)

6.4.2 Wake transition

It is well established in literature that a transition in the wake occurs at $Re \approx 212$ [6, 8, 17]. For a Reynolds number lower than this value, the wake is expected to be separated but still axis-symmetric, with a ring vortex that forms in the rear part of the sphere. For higher values, both experimental and numerical simulations confirm the break up of the axis-symmetrical wake. In both cases, the flow remains steady for Re up to approximately 270 [6, 17], for which the hairpin vortices start to periodically shed from the sphere [17]. However, for $212 < Re < 270$, a planar symmetry is still preserved, with a symmetry plane that assumes an arbitrary orientation with respect the flow field. The changes in the flow field also causes a very different force generated on the surface of the sphere. In particular, when the axis-symmetry breaks up, the drag is no more the only force which acts on the body and a lift contribution becomes quite pronounced. The direction of the positive lift contribution depends on the orientation of the symmetry plane.

Here, the numerical scheme is tested with two different Reynolds numbers, $Re = [200, 250]$. For $Re = 200$, a

separated, axis-symmetric wake is expected and only a drag contribution must be measured. In the other case, the transition to a steady, non-axisymmetric flow is expected. For $Re = 250$, the departure from the axialsymmetrical field is quite pronounced, as well as the forces generated by the flow on the sphere. The lift coefficient is no more null and it is used as an indicator to compare the solution with the available literature [6, 8, 17]. The computational domain adopted is a box $\Omega = [-L_1, L_2] \times [-L_3, L_3] \times [-L_3, L_3]$, with $L_1 = 10$, $L_2 = 30$ and $L_3 = 10$. The same boundary conditions of the previous tests are used for $\partial\Omega_f$ and for the IB surface. Hence, an inlet $\mathbf{u}_\infty = [1.0, 0.0, 0.0]$ is chosen for $x = -L_1$ and an outlet for $x = L_2$. Symmetry boundary conditions are adopted for the other faces of the computational domain. The immersed sphere is centred in the point $C = (0.0, 0.0, 0.0)$, it has a radius $R = 0.5$ and it is discretized by 1280 triangular faces. A *no-slip* boundary condition is assigned for the immersed surface. The computational domain is equally discretized for the directions y and z and it has a different discretization along the x direction, with an overall number of points equal to $256 \times 144 \times 144$. Let us consider y and z direction, the grid is built with a uniform interval size region for $[-R, R]$, then the interval size grows with a constant expansion ratio. For the x direction, the region $[-R, 4R]$ has a constant interval size, then two different expansion ratios are used for $x < R$ and $x > 4R$. For all the three directions, the uniform region around the sphere has an interval size of 0.02, which is equivalent to 50 points on the diameter.

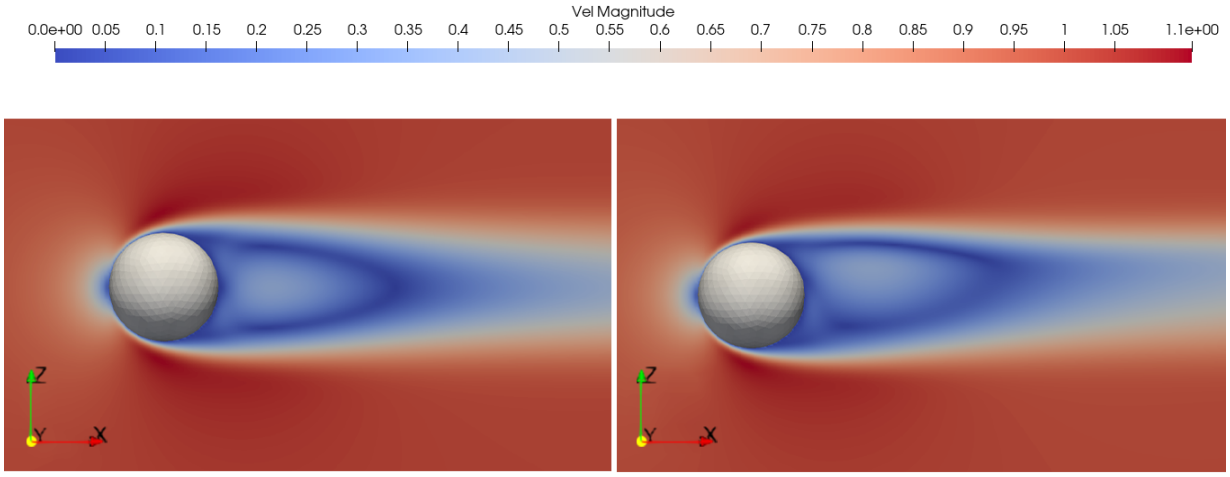
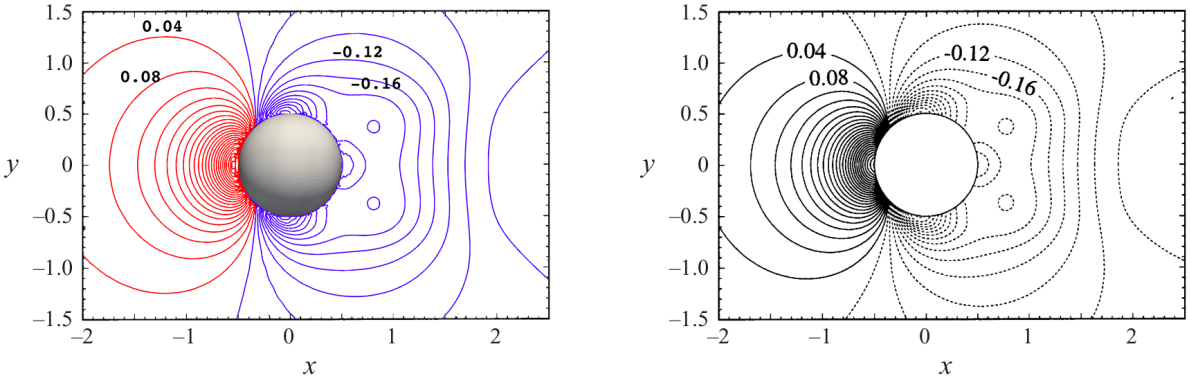


Figure 24: Velocity magnitude in the plane $y = 0$. *Left*: $Re = 200$. *Right*: $Re = 250$.



(a) IBM
Red lines: positive values, blue lines: negative values

(b) Johnson & Patel [17]
Continuous lines: positive values, dashed lines: negative values

Figure 25: Flow past a sphere at Re = 200. Pressure coefficient. Slice of the solution in the plane z=0. Comparison of the proposed method (*left*) with literature results(*right*).

In Figure 25, a comparison between the IBM and the numerical results of Johnson & Patel [17] is provided for $Re = 200$. In particular, the contour of the pressure coefficient $C_p = (p - p_\infty)/(0.5\rho||\mathbf{u}_\infty||^2)$ on the plane $z = 0$ is shown, with an increment of 0.04 of the C_p contours. The comparison shows a good agreement of

the numerical results obtained with the IBM with respect to the literature. Indeed, for example, both images provides indications about the local pressure minimum, identified by the closed circle behind the sphere, which corresponds to the core of the toroidal vortex in the recirculating wake. Such ring vortex structure behind the sphere is shown by the computation of the streamlines in the wake in Figure 26. Moreover, the measured drag coeffcient is $C_d = 0.769$, which is in agreement with the literature results of $C_d = 0.771$ in [6]. Another aspect

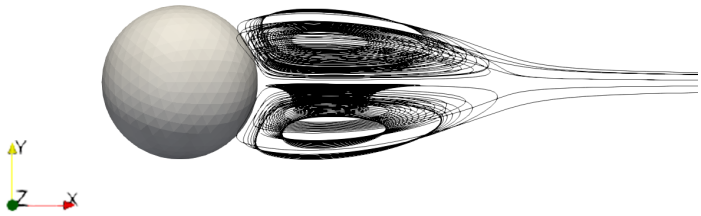


Figure 26: 3D Streamlines. *Re = 200*

which is used to compare the two results is the position of the stagnation point behind the sphere. Indeed, once the flow is separated, a toroidal or ring vortex structure forms in the wake and it closes in a point which is in a symmetrical position behind the sphere at a precise distance, namely the separation length x_s. In Figure 27, the x-component of the velocity over a line parallel to the x direction and passing through the center of the sphere is shown. The negative values in the wake of the sphere testify the presence of a recirculation region and the rear stagnation point occurs when the x-component of the velocity is null. The separation length is $x_s = 1.58$, which is quite in agreement with the literature results $x_s = 1.5$ in [6, 17].

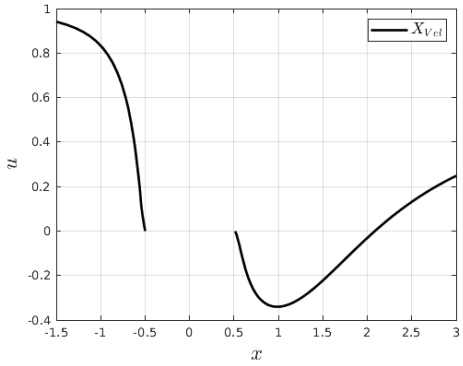
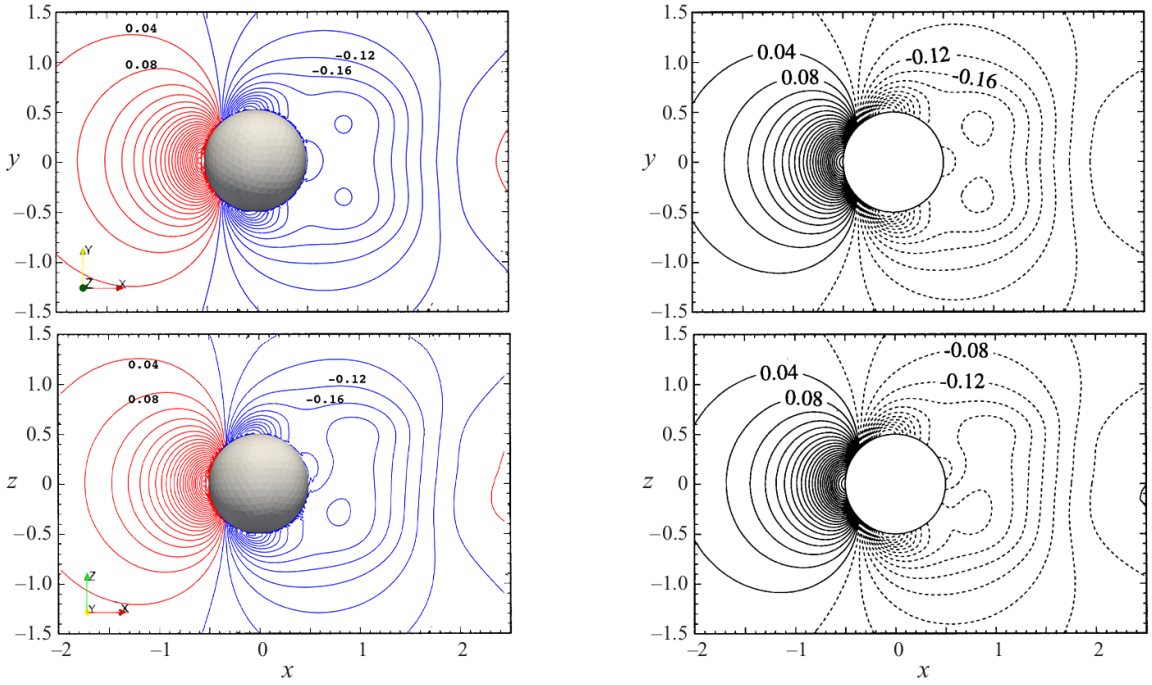


Figure 27: Flow past a sphere, Re=200. X-component of the velocity. Plot over line from (-1.5, 0.0, 0.0) to (3.0, 0.0, 0.0).

The second case considered corresponds to $Re = 250$. For such value, the axial symmetry of the flow breaks up and the flow exhibits a steady, planar symmetrical flow. A significant lateral force contribution is generated, with a direction that depends on the symmetry plane orientation which is arbitrary. Here, the symmetry plane corresponds to the x-z plane. The results are shown considering both slices on the symmetry x-z plane and on the x-y plane, to better evaluate the departure from the axial symmetric case previously described, and to compare it with the numerical results of Johnson & Patel [17]. Let us consider the numerical results on the two slices in Figure 28. In the x-y plane, the symmetry is noticeable for the pressure coefficient contours. Such situation is quite similar to the case of $Re = 200$. In contrast, in the numerical solution on the symmetry x-z plane, it is possible to point out the lost of axial symmetry in the recirculation region. In particular, the upper and lower vortical structures are no longer identical and the rear reattachment point is shifted upwards with respect to the case of $Re = 200$ (see Figure 24). This aspect is also evident by the analysis of the three-dimensional streamlines of the recirculation area to better capture the such region, in Figure 29. The toroidal structure has tilted to form a more complex separation region which preserves a planar symmetry, but it is also moved upwards. The loss of symmetry is noticeable also from the isocontours of the pressure coefficient in Figure 28. Here, symmetry is still in the x-y slice, whereas the solution on the symmetry plane reveals the higher pressure in the upper side with respect to the lower one which explains the lift force contribution. In the reference frame considered, the lift is directed as the z axis with a negative contribution. The results for the drag coefficient is $C_d = 0.703$ which is confirmed by the result of $C_d = 0.704$ in [6], whereas the measured lift coefficient is $C_l = -0.0559$ which has a larger error with respect to the other coefficients from the literature

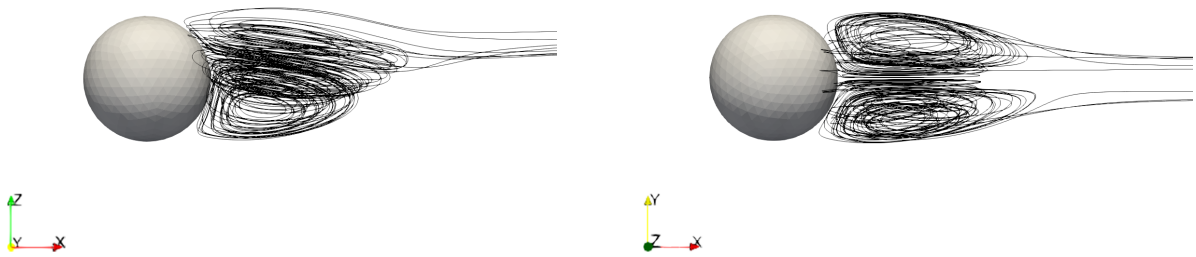
result of $C_l = -0.0610$ in [6] and $C_l = -0.0616$ in [8]. However, by such results, it is possible to confirm that the proposed IBM method correctly predict the transition to the planar symmetry. Moreover, the force computation, which is guaranteed to be only first order accurate, requires finer discretization for the sphere mesh which consequently requires, by construction, a finer Eulerian grid with a significant increase in terms of computational cost.



(a) IBM
Red lines: positive values, blue lines: negative values

(b) Johnson & Patel [17]
Continuous lines: positive values, dashed lines: negative values

Figure 28: Flow past a sphere at Re = 250. Pressure coefficient. Slice of the solution in the x-y (up) and x-z plane (down). Comparison of IBM with literature results.



(a) x-z plane

(b) x-y plane

Figure 29: Flow past a sphere at Re = 250. 3D Streamlines.

7. Conclusions

In this thesis a new Immersed Boundary approach is proposed. The enforcement of the velocity boundary condition on the immersed surface is done with interpolation techniques on a one-dimensional problem exploiting the direction-splitting scheme introduced by *Guermond and Minev* [16] and developed on a co-located grid by

Chiarini et al. [7]. Linear equations which discretize the boundary conditions are introduced preserving the banded tridiagonal structure of the problem to allow an efficient solution phase with the Thomas algorithm. The extrapolation of the solution into the solid domain allows one to correctly define a significant time history for all the variables that can potentially enter into the fluid domain avoiding the formation of numerical oscillations due to the fluid-solid discontinuity across the immersed boundary. The accuracy of the method is preliminarily tested in the one-dimensional diffusion equation by which the adopted direction-splitting scheme solves the incompressible Navier-Stokes problem. Convergence analyses are reported with both a fixed and a moving domain, demonstrating that the method preserves the second order accuracy of the finite-difference discretization. In the moving immersed boundary context, no source of spurious oscillations is present and no severe time step restriction is introduced making this method of interest for the simulation of moving objects at low Reynolds numbers. Manufactured solutions are used to verify the convergence rate also for the three-dimensional diffusion equation and for the fully 3D Navier-Stokes equations demonstrating that the IB method does not alter the second order temporal and spatial convergence. A strategy to compute the local stress acting on the IB surface is introduced and its accuracy tested with an analytical solution. Then, the whole IB method with such force computation scheme is tested using the flow past a sphere at varying Reynolds number as a benchmark to check the accuracy of the proposed method on a realistic flow. This work sets a step advancement into the simulation of low Re fluid dynamics problems in complicated, moving 3D geometries that are of interest in many fields. The account of a three-dimensional object into a fixed computational domain is now possible in the DNS code previously available, with good performance in terms of strong scalability. It is possible to move on towards cases of a moving object that could be, e.g, the imposed motion of the rotating propellers of small drones and the flapping wing movement of tiny insects that could be of interest in the bio-inspired design of FW-MAV. Moreover, it is possible to study fluid-structure interaction (FSI) problems. The great advantage in the grid generation and in the treatment of moving bodies seems to guarantee significant benefits in the simulation of flexible bodies. Further steps are needed in this path. A structural model must be introduced to treat a deformable object and the Navier-Stokes solver has to be coupled with the structural solver to simulate the fluid-structure interaction.

References

[1] S.A. Ansari, R. Żbikowski, and K. Knowles. Aerodynamic modelling of insect-like flapping flight for micro air vehicles. *Prog. Aerosp. Sci.*, 42(2):129–172, 2006.

[2] E. Balaras. Modeling complex boundaries using an external force field on fixed cartesian grids in large-eddy simulations. *Comput. Fluids*, 33(3):375–404, 2004.

[3] J. Barata, F. Neves, P. Manquinho, and T. Silva. Propulsion for Biological Inspired Micro-Air Vehicles (MAVs). *Open Journal of Applied Sciences*, pages 7–15, 2016.

[4] J. M. Birch and M. H. Dickinson. Spanwise flow and the attachment of the leading-edge vortex on insect wings. *Nature*, 412, 08 2001.

[5] J. M. Birch, W. B. Dickson, and M. H. Dickinson. Force production and flow structure of the leading edge vortex on flapping wings at high and low Reynolds numbers. *J. Exp. Biol.*, 207(7):1063–1072, 03 2004.

[6] G. Bouchet, M. Mebarek, and J. Dušek. Hydrodynamic forces acting on a rigid fixed sphere in early transitional regimes. *Eur. J. Mech. B. Fluids*, 25(3):321–336, 2006.

[7] A. Chiarini, M. Quadrio, and F. Auteri. A direction-splitting navier–stokes solver on co-located grids. *J. Comput. Phys.*, 429:110023, 2021.

[8] V. Citro, J. Tchoufag, D. Fabre, F. Giannetti, and P. Luchini. Linear stability and weakly nonlinear analysis of the flow past rotating spheres. *J. Fluid Mech.*, 807:62–86, 2016.

[9] M.D. de Tullio, R. Verzicco, and G. Iaccarino. Immersed boundary technique for large–eddy–simulation. *Lecture Notes, Von Karman Institute*, 2014.

[10] M. H. Dickinson and K. G. Götz. The wake dynamics and flight forces of the fruit fly Drosophila melanogaster. *J. Exp. Biol.*, 199(9):2085–2104, 09 1996.

[11] M. H. Dickinson, F. Lehmann, and S. P. Sane. Wing rotation and the aerodynamic basis of insect flight. *Science*, 284(5422):1954–1960, 1999.

[12] J. Douglas. Alternating direction methods for three space variables. *Numer. Math.*, 4:41–63, 1962.

[13] V. Eijkhout, R. van de Geijn, and E. Chow. Introduction to high performance scientific computing. 01 2016, https://web.corral.tacc.utexas.edu/CompEdu/pdf/stc/EijkhoutIntroToHPC.pdf.

[14] E.A. Fadlun, P. Orlandi, and J. Mohd-Yusof. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *J. Comput. Phys.*, 161:35–60, 06 2000.

[15] S. N. Fry, R. Sayaman, and M. H. Dickinson. The aerodynamics of hovering flight in Drosophila. *J. Exp. Biol.*, 208(12):2303–2318, 06 2005.

[16] J. Guermond and P-D. Minev. A new class of fractional step techniques for the incompressible navier–stokes equations using direction splitting. *Comptes Rendus Mathematique*, 348(9):581–585, 2010.

[17] T. A. Johnson and V. C. Patel. Flow past a sphere up to a reynolds number of 300. *J. Fluid Mech.*, 378:19–70, 1999.

[18] J. W. Keating. *Direction-Splitting Schemes For Particulate Flows.* PhD thesis, University of Alberta, Edmonton, Alberta, Canada, 2013.

[19] W. Kim and H. Choi. Immersed boundary methods for fluid-structure interaction: A review. *Int J Heat Fluid Flow*, 75:301–309, 2019.

[20] D. Lentink and Hawkes Elliot W. Fruit fly scale robots can hover longer with flapping wings than with spinning wings. *J. R. Soc.*, 2016.

[21] M.D. Mikhailov and A.P. Silva Freire. The drag coefficient of a sphere: An approximation using shanks transform. *Powder Technol.*, 237:432–435, 2013.

[22] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37(1):239–261, 2005.

[23] J. Mohd-Yusof. Combined immersed-boundary/b-spline methods for simulations of ow in complex geometries. *CTR, Annu Res Brief 1997*, pages 317–327, 1997.

[24] T. Möller and B. Trumbore. Fast, minimum storage ray/triangle intersection. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, page 7–es, New York, NY, USA, 2005. Association for Computing Machinery.

[25] L. W. Morland. A fixed domain method for diffusion with a moving boundary. *J. Eng. Math.*, 16:259–269, 1982.

[26] F. A. Morrison. *An Introduction to Fluid Mechanics.* Cambridge University Press, 2013.

[27] F. T. Muijres, M. J. Elzinga, J. M. Melis, and M. H. Dickinson. Flies evade looming targets by executing rapid visually directed banked turns. *Science*, 344(6180):172–177, 2014.

[28] P-Angot, J-Keating, and P. D. Minev. A direction splitting algorithm for incompressible flow in complex geometries. *Comput. Methods Appl. Mech. Eng.*, 217-220:111–120, 2012.

[29] C. S. Peskin. Flow patterns around heart valves: A numerical method. *J. Comput. Phys.*, 10(2):252–271, 1972.

[30] A. Quarteroni, R. Sacco, and F. Saleri. *Matematica Numerica.* Springer Milano, 2008.

[31] R. Ramamurti and W. C. Sandberg. A three-dimensional computational study of the aerodynamic mechanisms of insect flight. *J. Exp. Biol.*, 205(10):1507–1518, 05 2002.

[32] S. P Sane. The aerodynamics of insect flight. *J. Exp. Biol.*, 206:4191–208, 12 2003.

[33] W. Shyy, H. Aono, S.K. Chimakurthi, P. Trizila, C.-K. Kang, C.E.S. Cesnik, and H. Liu. Recent progress in flapping wing aerodynamics and aeroelasticity. *Progress in Aerospace Sciences*, 46(7):284–327, 2010.

[34] W. Shyy and H. Liu. Flapping wings and aerodynamic lift: The role of leading-edge vortices. *AIAA Journal*, 45(12):2817–2819, 2007.

[35] F. Sotiropoulos and X. Yang. Immersed boundary methods for simulating fluid–structure interaction. *Prog. Aerosp. Sci.*, 65:1–21, 2014.

[36] M. Sun and J. Tang. Unsteady aerodynamic force generation by a model fruit fly wing in flapping motion. *J. Exp. Biol.*, 205:55–70, 2002.

[37] S. Taneda. Experimental investigation of the wake behind a sphere at low reynolds numbers. *J. Phys. Soc. Jpn.*, 11(10):1104–1108, 1956.

[38] F. Tian, H. Dai, H. Luo, J. F. Doyle, and B. Rousseau. Fluid-structure interaction involving large deformations: 3d simulations and applications to biological systems with modeled wing motion. *J. Comput. Phys.*, 258, 2014.

[39] M. Uhlmann. An immersed boundary method with direct forcing for the simulation of particulate flows. *J. Comput. Phys.*, 209(2):448–476, 2005.

[40] S. Vogel. Flight in drosophila i. flight performance of tethered flies. *J. Exp. Biol.*, 44, 06 1966.

[41] S. Walker, A. Thomas, and G. Taylor. Deformable wing kinematics in free-flying hoverflies. *J. R. Soc. Interface*, 7:131–42, 06 2009.

[42] S. Xu and Z. J. Wang. A 3d immersed interface method for fluid–solid interaction. *Comput Methods Appl Mech Eng*, 197(25):2068–2086, 2008. Immersed Boundary Method and Its Extensions.

[43] J. Yang. Sharp interface direct forcing immersed boundary methods: A summary of some algorithms and applications. *J Hydrodynam B*, 28(5):713–730, 2016.

[44] J. Yang and E. Balaras. An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries. *J. Comput. Phys.*, 215(1):12–40, 2006.

[45] J. Yang and F. Stern. A simple and efficient direct forcing immersed boundary framework for fluid–structure interactions. *J. Comput. Phys.*, 231(15):5029–5061, 2012.

Abstract in lingua italiana

In questa tesi un metodo Immersed Boundary (IBM) viene sviluppato partendo da un codice DNS a differenze finite. L'IBM viene introdotto all'interno di uno schema direction-splitting usato per la risoluzione del modello di Navier-Stokes incomprimibile e le condizioni al contorno sul corpo immerso sono imposte con un approccio basato su tecniche di interpolazione ed estrapolazione. Le equazioni che impongono le condizioni al contorno sul corpo immerso sono inserite preservando la struttura a banda del sistema originale, permettendo una fase di risoluzione del sistema estremamente efficiente. Altri dettagli sulla parallelizzazione dei solutori e del trattamento dell'IB sono presentati. La presenza di un contorno immerso è trattata implicitamente all'interno del processo di risoluzione del sistema, risolvendo allo stesso tempo le equazioni di Navier-Stokes e le equazioni che discretizzano la condizione al contorno sull'IB, rendendo questo metodo interessante per simulazioni di problemi fluidodinamici a bassi numeri di Reynolds. L'efficacia e l'accuratezza del metodo proposto è stata testata sia su contorni immersi fissi che in movimento attraverso il confronto con soluzioni analitiche. Un metodo per il calcolo delle forze che agiscono localmente sul corpo viene proposto e testato con una soluzione analitica di riferimento e simulando il flusso attorno ad una sfera immersa in una corrente uniforme.

Parole chiave: Metodo Immersed Boundary, Differenze finite, Navier-Stokes incomprimibile

Acknowledgements

I would like to thank my thesis advisor, Professor *Franco Auteri*, for his inspiring lectures, his careful guidance and his time spent during the development of this thesis. In this period, through his experience, his advices and tips, he has been a reference figure always available for an open dialogue. I would also like to express my gratitude to *Alessandro Chiarini* for his availability and help.

Thanks to my parents, *Delia* and *Lauro*, for your support and for always trusting me in every occasion. Your emotional presence in facing every obstacle, even the most toughest one in appearance, has always made it easier to overcome. Thanks to my brother, *Lorenzo*, to always be at my side and to be my guide since we were younger.

To conclude, I would like to thank each person I met in this experience at Politecnico di Milano. I would like to thank *Francesco* and *Lorenzo* for helping me from the first days in this new path. Thanks to all my colleagues, *Enrico*, *Marco*, *Tommaso*, *Carlo*, with whom it was a pleasure to spend time together. I would like to spend few words to thank *Lorenzo* with whom I shared a good part of my university career. You have been a fantastic person, always ready to discuss and bring original ideas when we were working to a new project but, above all, able to make every situation fun, even the toughest ones. I would like to thanks *Fabrizio*, you are a good friend of mine and I will miss some moments spent together in these years.

Last but not least, thanks to my roommates, *Andrea*, *Matteo* and *Riccardo*, for the shared experiences in these years and for the fun of every moment spent together. From day one, you have been excellent companions in this journey, sharing daily joys and worries. You are the best I could ever meet!