Projection algorithm for simulation of fluid flow around moving objects with immersed boundary method

Alessia Abbati*, Andrea Chierici*, Leonardo Chirco*, Roberto Da Vià† and Sandro Manservisi*
*University of Bologna, DIN, Lab. of Montecuccolino, Via dei Colli 16, Bologna 40136, Italy
E-mail: †roberto.davia2@unibo.it

Abstract. The simulations of fluid flow around moving objects are usually done using body conforming meshes, but recently there is growing interest in algorithms using non conforming grids, known as immersed boundary methods (IB). In particular the resolution of the flow around objects with moving boundary may involve a high meshing work that has to be repeated at each time step. In the present work the immersed boundary method is applied to an in-house finite element code, in order to simulate transient applications involving the movement of immersed solid interfaces. The distinguishing feature of this method is that the entire simulation is carried out on a grid which cover the entire domain and do not conform to the geometry of the rigid objects. A second mesh independent from the first one and built on the immersed body is considered. The presented method consists in the projection of fields, including boundary conditions, from the second mesh mentioned to the first one, that cover the whole domain. The projection is carried out considering the movement of the immersed objects, through the handling of the second mesh. The proposed algorithm is built in order to satisfy the mass conservation in the entire fluid region and to avoid any numerical instability. For the projection of such fields we develop a multiphysics open-source coupling platform, based on the open-source SALOME platform. MED (Modèle d’Échange de Données) data structure has been used for all field operations, allowing the algorithm to be used with all the numerical codes where a MED duplicate of the field solution can be created. Applications of this approach will be presented and we provide results that show the consistency of the proposed algorithm.

1. Introduction
Over the last decades, numerical simulation has become an increasingly widespread tool to investigate complex engineering applications. Numerous computational fluid dynamics (CFD) problems involve fluid flow around moving objects within the framework of finite element approaches. Such systems are characterized by internal boundaries between fluid and solid regions with dynamically changing geometry. In this regard, one of the challenges to devising numerical techniques is to track the moving boundary and simulate its evolving interaction with the system. Even though considerable efforts have been made, simulating moving boundaries is still not a trivial problem. Methods for handling such problems can broadly be classified into Lagrangian, Eulerian and combined formulations according to their computational grid representation. Lagrangian methods feature a moving computational grid that is dynamically adapted to the interface. Boundary conditions can be applied at the exact location of the
interface since its evolution is explicitly tracked. These methods are widely used Fluid Structure Interaction problems solved with monolithic approach, allowing a better treatment of the stress continuity along solid-fluid interface [1, 2], for example in [3], where Fluid Structure Interaction problem is used to perform a preliminary investigation of a ventricle contraction, or [4] where a dual grid approach is used to couple an FSI problem with multiphase advection. The major drawback to this approach is the continuous grid rearrangement over the changing computational domain. Moreover large displacements are difficult to tackle since mesh element deformations can lead to stability issues due to excessive mesh element stretching. Conversely, Eulerian methods do not track the interface explicitly but reconstruct it by means of appropriate field variables. This allows a fixed grid formulation where the interface is not treated as a sharp discontinuity, which implies smearing of boundary information. In Eulerian methods topological changes are easier to handle since they do not require mesh rearrangement work and thus they are particularly suited for free surfaces applications. These methods are widely used in the case of multiphase flows [5, 6] and multiphase flows with solid particles, for example in [7], where an innovative dual grid approach is used to couple the multiphase CFD problem to the Discrete Element Method one. It is clear that for these applications an adaptive mesh algorithm cannot be used. The present work specifically address rigid boundaries problems by introducing a combined approach that aims to overcome the drawbacks resulting from both Eulerian and Lagrangian formulations.

Recent years have seen a significant interest in solution algorithms for fluid flow around objects based on non-conforming grids, namely meshes that are not configured to conform to the fluid-solid interface. Several numerical models referred to as immersed boundary methods (IB) have received attention due to their remarkable advantages in simulating fluid flow around moving objects with complex boundaries. The IB method was originally introduced by Peskin as a means of investigating cardiovascular flow patterns [8]. Although originally intended for biological fluid dynamics, the IB method has been successfully applied to a wide range of CFD problems and some noteworthy works have been proposed in the literature [9, 10]. Such methods share as a common core feature the ability to simulate the presence of boundaries without fitting the computational grid to the immersed body. The grid generation process is not altered and spatial discretization is performed over the whole computational domain, resulting in mesh elements being cut by the fluid-solid interface. IB methods constitute a promising alternative to conventional body-conformal grid approaches. The process of generating body-fitted grids requires extensive meshing work that may become prohibitively expensive for complex geometries. In order to handle moving boundaries problems, the grid needs to be regenerated at each time step, adding up to the computational costs. By contrast, IB methods allows simplified spatial discretization that results in more uniform computational grids obviating the need for mesh rearrangement. Because tackling complex moving boundaries is easier, non-conforming grid approaches experience less deterioration in grid quality which affects accuracy and convergence properties of the numerical scheme. Another common point of concern underlying IB methods is the imposition of boundary conditions on the fluid-solid interface. Conventional body-conformal meshes allows boundary conditions to be specified directly on the interface grid points with relative ease. Implementation of boundary condition is not as straightforward in IB methods, which in turn may negatively impact the robustness and conservation properties of the solution algorithm. IB methods account for the fluid-solid interface and its effects through several different strategies, which are extensively reviewed in [11]. The presence of the immersed body is mimicked by means of additional source terms and modified boundary conditions which may be imposed both by indirect and direct techniques.

In the present work, we present a finite element based IB method intended for simulation of fluid flow around moving objects. Numerical solution of fluid flow is computed on an Eulerian grid which spans over the entire domain whereas moving boundaries are tracked by a Lagrangian
reference frame. This combined formulation employs a fixed unstructured grid in conjunction with a moving unstructured grid built exclusively on top of the immersed body. The solid mesh accounts for the movement of the object without altering the fixed computational grid and consequently simplifying the discretization process. Coupling between fixed and moving mesh is tackled by appropriate projections of fields and boundary conditions. Thus, effects of moving objects on fluid flow may be imposed through indirect means by introducing an adequate projection algorithm. The proposed IB method succeeds in decoupling the equations for the fluid domain from those imposed on the moving solid and consequently no numerical solution is required for governing equations inside the immersed body. The pressure discontinuity associated with the fluid-solid interface requires modifying the computational stencil near the immersed body to impose an additional boundary condition. Effects of additional parameters introduced in this formulation are not straightforward and may negatively affect computational stability and numerical solution conservation properties.

We develop a multiphysics coupling tool based on the open-source SALOME platform to implement advanced interpolation techniques [12, 13]. MED (Modèle d’Échange de Données) data structure has been used as a common format to define information exchange between codes. Manipulation of meshes and fields in numerical schemes involved in a simulation process is greatly simplified by adopting a common exchange format. Reduced code coupling complexity allows sharing advanced functions and preserving existing routines. Several interpolation methods have emerged in recent years and a detailed discussion on this issue is presented in [4]. Interpolation techniques projects numerical fields from a source computational grid to a target domain. In the proposed method we use consistent or node-wise interpolation and we evaluate its performance potential. A key factor in consistent interpolation is imposing equality between the node field values on the target domain and the corresponding source mesh. In this work we define a piece-wise indicator function on the support of the solid moving grid, which is set to be the source mesh. The algorithm then projects this field to the target fixed grid spanning over the entire computational domain in order to define a discretized indicator function. The resulting function allows to represent the immersed body and its location in the fluid domain. Additionally, we specify a cell-wise volume fraction field to evaluate the solid body portion of each computational cell. A different interpolation scheme, denoted Galerkin projection, is then implemented to determine a point-wise target field. This information allows to explicitly track the fluid-solid interface through linear reconstruction. The proposed method represent an efficient approach to interface reconstruction since it enables the computation of interface variables without being prohibitively complex and time consuming. Numerical solution for fluid flow around a moving object is then accurately computed on the fixed grid. At each time iteration the solid computational grid position is changed using an arbitrary law, involving a fixed movement or an interaction with fluid through stress calculation on fluid-solid interface. Consequently indicator function is interpolated and the volume fraction field updated on the target mesh (fluid computational grid). Numerical solution accuracy strictly relates to the robustness of the projection algorithm, thus it is worthwhile to investigate the interpolation process. We assess the performance of the proposed IB approach with two significant test cases with the intent of validating the numerical scheme.

2. Numerical Modeling

Let us consider a moving rigid body $\Omega_s(t)$ completely immersed in a fluid domain $\Omega_f(t)$ and separated by a time dependent interface $\Gamma_{f,s}(t) = \partial \Omega_f(t) \cap \partial \Omega_s(t)$, as sketched in Figure 1 a). Any topological changes in the solid region affect in turn the fluid domain $\Omega_f(t)$ in such a way that the entire computational domain $\Omega = \Omega_f(t) \cup \Omega_s(t)$ does not vary over time. The governing equation considered here is the unsteady Navier-Stokes equation for incompressible laminar flow.
Figure 1. Sketch representation a) of a computational domain made of a fluid (\(\Omega_f\)) and solid (\(\Omega_s\)) sub domains which interact through a fluid-solid interface (\(\Gamma_{f,s}\)) and close-up view b) of computational grid cut by the fluid-solid interface.

given by, namely

\[
\begin{align*}
\nabla \cdot \mathbf{u} &= 0, \\
\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} &= -\frac{1}{\rho} \nabla P + \nu \nabla^2 \mathbf{u} + \mathbf{g}.
\end{align*}
\]

where \(\mathbf{u}\) is the velocity vector, \(\rho\) the density, \(P\) the pressure, \(\nu\) the kinematic viscosity and \(\mathbf{g}\) the gravity acceleration vector.

2.1. Field projection

Let us introduce a source field \(\Psi_s\) defined on the computational domain \(\Omega_s\) and a target field \(\Psi_t\) in \(\Omega_t\). Let \(N_e^s\) be the number of elements and \(n_e\) the number of nodes that compose \(\Omega_s\). The projection operator that interpolates the source field \(\Psi_s\) to \(\Psi_t\) is defined by

\[
\Psi_t = \prod_{\Omega_t} (\Psi_s) = \sum_{e=1}^{N_e^s} \sum_{j=1}^{n_e} \Psi_{s,j}^e \varphi_j^e (\pi^{-1}(x_{t,j}^e)).
\]

where \(\varphi_j^e\) are the approximation functions, \(\pi^{-1}\) the linear transformation from global coordinates to canonical ones and \(x_{t,j}^e\) the target field node coordinates. The projection operator may be denoted by \(|P|\) and written in the following compact form

\[
\Psi_t = |P|\Psi_s
\]

More specifically, the proposed projection scheme implements in turn a search point algorithm, reverse mapping and an interpolation algorithm. A more detailed description of the point-wise interpolation method can be found in [4].

2.2. Immersed boundary formulation

In the present formulation of Immersed Boundary algorithm we will label with \(\Omega_s(t)\) and \(\Omega_f(t)\) the solid and fluid physical regions, respectively, as sketched in Fig. 1 a). The union of these two regions will be denoted as \(\Omega\). We remark that \(\Omega\) is not a function of time and that this region will be used as computational domain. When dealing with immersed boundary algorithms it is important to define a method for identifying fluid and solid regions. In literature many methods
Figure 2. Example of discretized indicator function $\chi^h$ a) of volume fraction $\alpha$, together with a mesh representation of solid body b), and interpolated volume fraction $\alpha'$ together with a fluid-solid interface reconstruction c).

Figure 3. Close-up view of interface reconstruction on a generic mesh element.

are proposed, involving, for example, level set methods based on wall distance between a mesh node and the fluid-solid interface [9, 10]. In the present work we adopt an indicator function $\chi(x, t)$ defined as a multidimensional Heaviside function, meaning

$$\chi(x, t) = \int_{\Omega_s(t)} \delta(x' - x) \, dx' \quad \forall x \in \Omega,$$

(5)

where $\delta$ is the Dirac delta function. The possible values of $\chi(x, t)$ over $\Omega$ are then

$$\chi(x, t) = \begin{cases} 1, & \forall x \in \Omega_s(t) \\ 0, & \forall x \in \Omega_f(t) \end{cases}.$$  

(6)

A discretized indicator function $\chi^h(x, t)$ is computed using the projection method described in Section 2.1. In particular we define a source field $\psi_s$ on the solid body mesh, with a uniform value $\psi_s(x) = 1 \forall x \in \Omega_s$. The field $\chi^h(x, t)$ is then calculated as a projection of $\psi_s$ over the computational domain, meaning $\chi^h = |P|\psi_s$. Alongside the point-wise indicator function $\chi^h$ we define a cell-wise volume fraction field $\alpha$ that, for each computational cell, indicates the fraction of volume occupied by the solid body. Examples of both $\chi^h$ and $\alpha$ are reported in Fig. 2 a) and b), respectively. We finally introduce a point-wise field, denoted as $\alpha'$, which is calculated with a Galerkin projection of volume fraction $\alpha$, namely

$$\int_{\Omega} (\alpha' \phi + \gamma \nabla \alpha' \cdot \nabla \phi) \, d\Omega = \int_{\Omega} \alpha \phi \, d\Omega \quad \forall \phi \in H^1(\Omega),$$  

(7)
where $\gamma$ is an arbitrary diffusion coefficient. Field $\alpha'$ can be used to perform a linear reconstruction of real solid-fluid interface $\Gamma_{f,s}$ as labeled in Fig. 1 a) and b). The interpolated values of the $\alpha$ field shown in Fig. 2 b) are reported in Fig. 2 c), together with the interface linear reconstruction. The linear reconstruction is performed as follows. As presented in Fig. 3, a generic mesh cell, labeled as $\Omega_j$, having a volume fraction value $0 < \alpha_j < 1$, can be divided into sub-elements $\Omega^f_j$ and $\Omega^s_j$ by means of a straight edge. The interface normal unit vector $\hat{n}$ is calculated as the ratio of local $\alpha'$ field gradient and the gradient modulus. The interface edge $\Gamma_{sl}^j$ inclination is computed in accordance to $\hat{n}$, while its position is determined from an area conservation equation, namely

$$\int_{\Omega^s_j} d\Omega = \alpha_j \int_{\Omega_j} d\Omega. \quad (8)$$

The reconstructed interface can be used for evaluating quantities of interest, like the stress exerted from the fluid on solid body surface.

The weak formulation of the solved system of equation is then

$$\int_\Omega \frac{\partial u}{\partial t} \cdot \varphi d\Omega + \int_\Omega (u \cdot \nabla) u \cdot \varphi d\Omega =$$

$$= -\frac{1}{\rho} \int_\Omega \nabla P \cdot \varphi d\Omega + \int_\Omega \nu(\nabla^2 u) \cdot \varphi d\Omega + \int_\Omega g \cdot \varphi d\Omega, \quad \forall \varphi \in H^1_0(\Omega) \quad (9)$$

$$\left\{ \begin{array}{ll}
\int_{\Omega_j} (\nabla \cdot u) \psi d\Omega = 0, & \forall \psi \in L^2(\Omega) & \forall \Omega_j : \alpha_j < \tilde{\alpha} \\\n\int_{\Omega_j} (\lambda \nabla^2 P - \nabla \cdot u) \psi d\Omega = 0, & \forall \psi \in L^2(\Omega) & \forall \Omega_j : \alpha_j > \tilde{\alpha} \end{array} \right. \quad (10)$$

$$u_i = \tilde{u}_s \quad \forall x_i : \chi(x_i, t) = 1, \quad (11)$$

where $\lambda$ is a constant value and $\tilde{\alpha}$ is a threshold value. In cells where $\alpha > \tilde{\alpha}$ we solve a pressure equation that is similar to the ones used in velocity projection methods [14]. It will be later shown that this equation allows to obtain a smoother pressure field. As reported in (11), the fluid velocity field in the solid-occupied computational region is modeled with the solid velocity field. In particular, the values $\tilde{u}_s$ imposed in (11) are obtained by a point-wise projection of the velocity field $u_s$ that is defined on the solid body mesh, namely $\tilde{u}_s = \|P\| u_s$. As can be seen from the solved system of equations, in the present immersed boundary formulation we don’t use forcing terms for the imposition of boundary conditions along the solid-fluid interface. The algorithm used for the realization of the numerical simulation with the present immersed boundary method can be summarized, for a generic $n$-th time step, as:

(i) Solid position on computational grid: creation of volume fraction $\alpha$, projection of solid velocity $\tilde{u}_s$ and indicator function $\chi^h$

(ii) System of equations solution

(iii) Calculation of interpolated volume fraction $\alpha'$

(iv) Reconstruction of solid-fluid interface $\Gamma_{f,s}$

(v) Stress calculation

(vi) Update solid position and velocity field

As explained above, the present Immersed Boundary method is formulated similarly to a Volume Of Fluid method for multiphase problems since in both cases an indicator function and a volume fraction field are used to distinguish one phase from another and to reconstruct the interface between the two phases. Differently from the multiphase problem, where the evolution of the indicator function is studied with the solution of an appropriate advection equation, the
simulation of non deformable solid bodies allows to compute the indicator function field with a projection algorithm. With the projection operator arbitrarily moving solid objects can be handled without the need of computing a wall distance field between mesh nodes and the fluid-solid interface, which could be a cumbersome task when dealing with complex solid shapes [9, 15]. Moreover, if the solid body position is fixed in time, in the above described algorithm steps (i), (iii), (iv) and (vi) can be performed just once.

The present Immersed Boundary method has been implemented into Finite Element code FEMuS [16].

3. Results
In the present section we describe the results obtained for two different test cases, involving both a fixed and a moving obstacle.

3.1. Flow around a cylinder at Re=20

\[ U = 0 \]

\[ a \quad b \quad c \quad d \quad L \]

\[
\begin{array}{cccc}
   a [m] & b [m] & c [m] & d [m] & L [m] \\
0.21  & 0.2   & 0.2   & 0.1  & 2.2  \\
\end{array}
\]

Table 1. Geometrical parameters of the simulated domain.

\[
\begin{array}{ccc}
   \text{REF0} & \text{REF1} & \text{REF2} \\
   d/\delta x = d/\delta y & 7.2 & 14.5 & 29 \\
   \varepsilon & -1.5 \times 10^{-5} & -6 \times 10^{-6} & -1.4 \times 10^{-6} \\
\end{array}
\]

Table 2. Grid resolution and mass loss coefficient for the three different used computational grids.

The present test consists of simulating a stationary fluid flow around a fixed cylinder. This benchmark case is well studied in [17]. Let \( U_0 \), \( d \) and \( \nu \) be the mean inlet velocity, the cylinder diameter and the fluid kinematic viscosity, respectively. The resulting Reynolds number is \( Re = U_0d/\nu = 20 \). A sketch of the simulated domain, together with the imposed boundary conditions, is reported in Fig. 4, while the geometrical parameters are described in Tab. 1, where \( L \) is the length of the channel. A parabolic velocity profile is imposed on the inlet section, namely

\[
U(x, y) = 4U_{max} \frac{(y_{max} - y)(y - y_{min})}{(y_{max} - y_{min})^2},
\]

(12)
where \( y_{\text{max}} \) and \( y_{\text{min}} \) are the channel maximum and minimum \( y \) coordinate values, and \( U_{\text{max}} \) is the maximum velocity value, \( U_{\text{max}} = 1.5U_0 \). One major concern when dealing with immersed boundary methods is mass conservation. For the present case we use three different computational grids with increasing levels of mesh refinements, in order to demonstrate that the proposed method allows to obtain a good mass conservation. The three different meshes are parameterized as a function of grid resolution with respect to the obstacle diameter and will be hereafter labeled as \( \text{REF0}, \text{REF1} \) and \( \text{REF2} \), where the number stands for the number of refinements. The grid resolution for the three different meshes is reported in Tab. 2 as a function of the ratio \( d/\delta x \), where \( d \) is the obstacle diameter and \( \delta x \) is the mesh size along \( x \) direction.

In order to evaluate the behavior of the proposed immersed boundary method regarding mass conservation, we report the values of mass loss fraction \( \varepsilon \) defined as \( (U_0 - U_{m,\text{out}})/U_0 \), where \( U_{m,\text{out}} \) is the mean outlet velocity. As can be seen from the values reported in Tab. 2, the mean outlet velocity is slightly higher than the mean inlet velocity value and mass loss fraction decreases by an order of magnitude using a double mesh refinement. In the present test we studied the influence of the diffusion coefficient \( \lambda \) that is used in (10) for the pressure equation in the solid-occupied cells. In Fig. 5 we report a set of close-up views of the pressure fields obtained with the three different mesh refinements and with three different values of \( \lambda \), namely \( 10^{-5}, 10^{-3} \) and \( 10^{-1} \). We see that pressure spots arise in the vicinity of fluid-solid interface when using a small value of \( \lambda \). Although \( \lambda \) value has not a sensible impact on mass conservation (the values of Tab. 2 refer to \( \lambda = 10^{-3} \) but are quite the same for each of the tested values of \( \lambda \)), we see that \( \lambda \) has a great impact on fluid behavior in the near solid body region, as can be seen from the flow streamlines reported in Fig. 5. Finally we compare the obtained results for pressure difference \( \Delta P \) between front and rear side of the cylinder and drag coefficient \( c_d \) with
Table 3. Computed values of pressure difference $\Delta P$ between front and rear side of the cylinder and of drag coefficient $c_d$, together with reference data [17].

| Case  | $\Delta P$  | $c_d$  |
|-------|-------------|--------|
|       | $10^{-5}$   | $10^{-3}$ | $10^{-1}$ | $10^{-5}$ | $10^{-3}$ | $10^{-1}$ |
| REF0  | 0.1072      | 0.1122  | 0.0741    | 5.051    | 5.185    | 5.351     |
| REF1  | 0.1105      | 0.1116  | 0.0671    | 4.897    | 5.014    | 5.224     |
| REF2  | 0.1162      | 0.1120  | 0.0532    | 4.912    | 5.019    | 5.060     |
| Reference | 0.1175 | 5.579 |

The reference ones, discussed in [17]. The computed values for all simulated cases, in terms of computational grid and $\lambda$ values, are reported in Tab. 3 along with reference values. The drag coefficient is here calculated as

$$c_d = \frac{2F_x}{\rho dU_0^2} \rightarrow F = \int_S \left[ \mu \left( \nabla u + \nabla u^T \right) - pI \right] \cdot \hat{n} dS,$$

where $\hat{n}$ is the normal to the solid-fluid interface, pointing from solid to fluid region, $S$ is the area of solid-fluid interface and $F_x$ the $x$ component of the stress $F$. In the present case the integral is calculated over the reconstructed interface. With small values of $\lambda$ we observe an underestimation of pressure difference, ranging from 1 to 8%, while with $\lambda = 0.1$ we obtain much higher errors (37 to 54%). If we exclude the case $\lambda = 0.1$, the percentage difference between the computed drag coefficient and the reference one lays in the range 7 to 12%.

3.2. Flow around an impulsively started cylinder

Figure 6. Sketch of the simulated domain for the case of flow around an impulsively started cylinder.

In the present test we simulate the case of a cylinder that is impulsively put in motion with a constant and uniform velocity $U_0$. For this particular case an experimental study was performed in [20], as well as numerical simulations with vortex method [18, 21, 19] and immersed boundary techniques [22, 23] for various Reynolds numbers. A sketch of the simulated domain is reported.
Figure 7. Comparison of drag coefficient evolution, for cases REF0, REF1 and REF2, with analytical law (dotted line) [18] and numerical reference values (dash-dotted line) [19].

Figure 8. Equi-vorticity contours for the three simulated cases at three different non-dimensional time steps. Vorticity values range from -3 to 3 with a step of 0.4.

in Fig. 6, together with parameters $a$, $b$ and $d$, denoting the domain width, height and cylinder diameter respectively. The computational box is characterized by the values $a = 12d$ and $b = 10d$. A no-slip boundary condition is imposed on all boundaries. The Reynolds number,
based on cylinder velocity modulus \( U_0 \), cylinder diameter and fluid kinematic viscosity, is equal to 40. At time \( t = 0 \) the cylinder is put into motion with a constant velocity, moving from right to left. The simulated time interval is \( t \in [0, 3.5\tilde{t}] \), where \( \tilde{t} = d/U_0 \). Numerical results will be presented as a function of non-dimensional time \( t^* = t/\tilde{t} \), in order to compare them with literature values [18, 19, 22, 23]. We use two different computational grids that are characterized by the following mesh sizes \( \delta x \) and \( \delta y \), along \( x \) and \( y \) direction: \( \delta x = \delta y = d/9 \) for case \( \text{REF0} \), \( \delta x = \delta y = d/18 \) for case \( \text{REF1} \) and \( \delta x = \delta y = d/36 \) for case \( \text{REF2} \). In Fig. 7 we report the time evolution of the drag coefficient \( c_d \) for the three simulated cases and compare them with the reference numerical results [19] and with the analytical relation proposed in [18], namely

\[
c_d = \frac{4\sqrt{\pi}}{\sqrt{t^* Re}} + \frac{\pi}{Re} \left( 9 - \frac{15}{\sqrt{\pi}} \right), \tag{14}
\]

which is valid for \( t^* < 0.5 \). As in the previous test, the drag coefficient is computed using (13) on the reconstructed solid-fluid interface. We observe that the computed values of \( c_d \) are highly oscillating for case \( \text{REF0} \), while for the other two cases oscillations amplitude decrease with mesh refinement. Drag coefficient values are slightly underestimated, with respect to the reference values [19], in the interval \( t^* \in [0, 2] \), while for higher values of the non-dimensional time we observe a good agreement. A comparison between the simulated cases of the obtained vorticity \( \omega \) values is reported in Fig. 8 for three different time steps, namely \( t^* = 0.5 \), \( t^* = 1.0 \), \( t^* = 2.5 \) and \( t^* = 3.5 \). In particular, non-dimensional vorticity \( \omega^* \) contours are shown in a range \( \omega^* \in [-3, 3] \) with increments of 0.4, in order to compare them with the reference results [22, 23]. Comparing case \( \text{REF0} \) and \( \text{REF1} \) we see that grid spacing has a major impact on the evaluation of vorticity in the fluid area behind the cylinder and also in the estimation of a higher vorticity region. An increase of mesh resolution from \( \text{REF1} \) to \( \text{REF2} \) leads to less considerable changes in the predicted values of \( \omega^* \). We observe a very good agreement with the reference profiles even though the maximum mesh resolution used here is much smaller, namely \( \delta x/d = 36 \) instead of \( \delta x/d = 50 \) and \( \delta x/d = 100 \) [22, 23]. As a final test we study the evolution of the recirculation length \( l^* \) and compare with reference data [20].

![Figure 9. Time evolution of recirculation length \( l^* \) and comparison with reference data [20].](image-url)

recirculation zone length \( l \). Non-dimensional values of this variable, \( l^* = l/d \), are plotted against
non-dimensional time $t^*$ in Fig. 9 for all the simulated cases and compared with reference values [20]. The length $l^*$ is calculated by examining the velocity field $U - U_0$, i.e. from a reference frame defined on the moving cylinder. For the first part of the simulated time interval, namely $t^* < 1$, we observe a good agreement between our results and the reference ones. For higher values of non-dimensional time the results obtained from case REF0 increasingly underestimated $l^*$, while the ones of case REF1 and REF2 are still close to reference data.

4. Conclusions
In the present paper we presented a simple immersed boundary method implementation. The proposed method is based on a computational grid for the fluid flow and one for the solid body. By using numerical fields projection, we can define a velocity and an indicator function field on the solid body mesh and then project them on the fluid computational grid in order to define solid-occupied regions and to calculate velocity values on those particular grid nodes. A key aspect of this procedure is that the method can be applied on arbitrarily shaped objects that can either be static or move. In mesh cells occupied by solid, velocity values are imposed with Dirichlet boundary condition while an equation similar to those used for Navier-Stokes projection methods [14] is solved for pressure, in order to obtain both mass conservation and a smooth pressure field. We report two test cases, involving respectively a static solid body and a moving one. In the first test we study mass conservation, pressure difference between front and rear sides of solid obstacle and drag coefficient values on a reconstructed solid-fluid interface. We use three different domain discretizations with increasing mesh refinements. The mass conservation error rapidly decreases with the increase of mesh resolution and also pressure difference is in good agreement with reference data [17]. Higher discrepancy between the present calculations and reference values is found for the drag coefficient. The second test involves a cylinder that is suddenly set in motion with uniform and constant velocity in a fluid that is initially quiescent. Even for this case we use three different domain discretizations and we study drag coefficient values, recirculation zone length and vorticity field. The results are compared with reference data [18, 21, 19, 22, 23] and a very good agreement is found. The results presented for both test cases are very promising and positively contribute to the validation of the numerical approach. Future developments will involve simulations with higher Reynolds number values and three-dimensional objects.

References
[1] Michler C, Hulshoff S, van Brummelen E and de Borst R 2004 Comput. Fluids 33 839 – 848
[2] Causin P, Gerbeau J and Nobile F 2005 Comput. Methods. Appl. Mech. Eng. 194 4506 – 4527
[3] Cerroni D, Giommi D, Manservisi S and Mengini F 2018 Preliminary monolithic fluid structure interaction model for ventricle contraction Biomedical Technology: Modeling, Experiments and Simulation ed Wriggers P and Lenarz T (Springer International Publishing) pp 217–231
[4] Cerroni D, Da Viá R and Manservisi S 2018 J. Comput. Phys. 354 646–671
[5] Rider W J and Kothe D B 1998 J. Comput. Phys. 141 112 – 152
[6] Tryggvason G, Scardovelli R and Zaleski S 2011 Direct numerical simulations of gas–liquid multiphase flows (Cambridge University Press)
[7] Pozzetti G and Peters B 2018 Int. J. Multiph. Flow 99 186 – 204
[8] Peskin C S 1972 J. Comput. Phys. 10 252–271
[9] Ilinca F and Hétu J F 2011 Int. J. Numer. Methods Fluids 65 856–875
[10] Jendoubi A, Yakoubi D, Fortin A and Tibirna C 2014 Int. J. Numer. Methods Fluids 75 63–80
[11] Mittal R and Iaccarino G 2005 Annu. Rev. Fluid Mech. 37 239–261
[12] Cerroni D 2015 Multiscale multiphysics coupling on a finite element platform Ph.D. thesis University of Bologna
[13] Chierici A, Chiro L, Da Viá R, Manservisi S and Scardovelli R 2017 J. Phys. Conf. Ser. 923 012025
[14] Guermond J, Minev P and Shen J 2006 Comput. Methods Appl. Mech. Eng. 195 6011 – 6045
[15] Hétu J F and Ilinca F 2013 Comput. Fluids 87 2 – 11
[16] Code FEMus URL https://github.com/FemusPlatform/femus/
[17] Schäfer M et al. 1996 Benchmark Computations of Laminar Flow Around a Cylinder (Wiesbaden: Vieweg+Teubner Verlag) pp 547–566
[18] Bar-Lev M and Yang H T 1975 J. Fluid Mech. 72 625–647
[19] Koumoutsakos P and Leonard A 1995 J. Fluid Mech. 296 1–38
[20] Coutanceau M and Bouard R 1977 J. Fluid Mech. 79 257–272
[21] Chloe M, Federico G, Georges-Henri C and Iraj M 2015 Int. J. Numer. Methods Fluids 79 55–83
[22] Taira K and Colonius T 2007 J. Comput. Phys. 225 2118 – 2137
[23] Shang-Gui C, Abdellatif O, Julien F and Yannick H 2017 Int. J. Numer. Methods Fluids 85 288–323