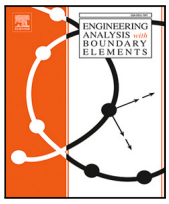
Engineering Analysis with Boundary Elements 179 (2025) 106345



Contents lists available at ScienceDirect

Engineering Analysis with Boundary Elements

journal homepage: www.elsevier.com/locate/enganabound





# Fast implicit direct-forcing immersed boundary method (FIDF-IBM)

Elias Farah [a,b,*], Abdellatif Ouahsine [a], Patrick Verdin [b]

[a] Centre de Recherche Royallieu, UMR 7337 Roberval, Université de Technologie de Compiègne, Alliance Sorbonne Université, Rue du Dr Schweitzer, Compiègne, 60200, France

[b] Faculty of Engineering and Applied Sciences, Cranfield University, College Rd, Wharley End, Cranfield, MK43 0AL, United Kingdom

ARTICLE INFO

Dataset link: https://doi.org/10.57996/cran.ceres-2760

*Keywords:*
Immersed boundary method
Lagrangian weight
Operator splitting
Incompressible flows
Finite volume method

ABSTRACT

A fast implicit direct-forcing immersed boundary method (FIDF-IBM) is introduced for the simulation of incompressible flows over arbitrarily moving solid structures. This method leverages the operator splitting approach of the pressure implicit with splitting of operators (PISO) algorithm to decouple the pressure, velocity, and boundary force in the solution process. This maintains the no-slip/no-penetration (ns/np) boundary constraint and enforces the divergence-free condition in a segregated manner in the solid and fluid domains, respectively. The proposed scheme produces a modified pressure Poisson equation (PPE) that includes the boundary force already satisfying the ns/np boundary constraint, allowing the usage of fast iterative PPE solvers. The term "fast direct-forcing" is achieved by coupling Lagrangian weight methods that enhance the reciprocity of the IBM-related linear operators with the IBM implicit formulation. Additionally, an appropriate boundary force inheritance from previous time-steps further boosts the performance of the implicit DF-IBM algorithm. The method's efficiency and capability are verified through different stationary and moving immersed boundary benchmark tests.

## 1. Introduction

Developed by Peskin in 1972 [1], the Immersed Boundary Method (IBM) employs two distinct domains: the Eulerian domain, where the fluid flow governing equations are solved, and the Lagrangian domain, which represents the solid immersed boundary using discrete Lagrangian markers. In this approach, interaction equations are developed, known as interpolation and spreading operators; they serve as the data mapping between the two domains. The novelty of IBM lies in introducing a fictitious body force into the governing fluid Navier–Stokes equations as a source term to account for the effects of the solid immersed boundary. This fictitious body force, ensures the desired velocity boundary condition on the Lagrangian markers, referred herein as the Eulerian boundary force. Based on how this force is treated, several review papers can be found [2–7], which discuss thoroughly and extensively all variants of the IBM and their possible area of applications along with a comprehensive analysis of the current developments in the IBM field.

For the present work, the direct-forcing IBM (DF-IBM) variant developed by Uhlmann [8] was employed, owing to its broad adoption by the IBM community, versatility, and straightforward integration into various fluid solvers. The originality of the DF-IBM lies in eliminating the constitutive relation restrictions for rigid bodies imposed by the stiffness coefficient of Hooke's law in the classical IBM [1], primarily designed for elastically deformable boundaries. Therefore, the kinematics are enforced directly on the immersed boundary, i.e., the no-slip/no-penetration (ns/np) boundary constraint, and the assumptions about the system's dynamics are removed. By doing so, we are penalizing the ns/np error at the fluid-solid interface, causing flow penetration into the solid boundary, because the ns/np condition is enforced using a predicted, non-divergence-free velocity field rather than the corrected flow field, which is unknown in the DF-IBM steps. This ns/np error is notorious for generating an undesirable and unphysical flow inside the immersed boundary, causing an overprediction of the aerodynamic coefficients as shown in [9,10].

The first explicit approach of Uhlmann [8], termed as the explicit DF-IBM, struggled to impose the ns/np condition on the immersed boundary, as highlighted in several studies [9,11,12]. To address this issue, a multi-direct forcing approach, termed multi-DF-IBM, was introduced. Previous studies [11,12] indicated that three loops are typically sufficient to minimize errors on the immersed boundary while maintaining computational efficiency. For exact ns/np constraint imposition, implicit IBMs were developed. A fully implicit IBM, known as the immersed boundary projection method (IBPM), was developed by Taira and Colonius in 2007 [13]. This method couples the velocity field

* Corresponding author at: Faculty of Engineering and Applied Sciences, Cranfield University, College Rd, Wharley End, Cranfield, MK43 0AL, United Kingdom.
*E-mail address:* elias.farah@cranfield.ac.uk (E. Farah).

https://doi.org/10.1016/j.enganabound.2025.106345
Received 25 November 2024; Received in revised form 3 June 2025; Accepted 6 June 2025
Available online 2 July 2025
0955-7997/© 2025 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

with both the pressure field and boundary force to enforce divergence-free and ns/np constraints using Lagrange multipliers. A new pressure Poisson equation (PPE) is formulated with pressure and boundary force as unknowns, solved simultaneously in a single projection step. However, Ji et al. [14] found that IBPM encounters convergence issues when a Lagrangian marker is near an Eulerian grid point, increasing the condition number of the PPE's coefficient matrix due to the interpolation and spreading operators' singularity. Ji et al. [14] resolved these convergence issues by iterating each constraint separately, maintaining an unmodified coefficient matrix and nesting the ns/np constraint within the continuity constraint, significantly increasing the algorithm's computational effort. To balance computational efficiency and accuracy, the Moving Implicit IBM (MIBM) was developed in 2017 [15]. MIBM introduces an additional equation for the boundary force, termed the moving force equation. Similar to the IBPM approach [13], MIBM treats the boundary force as a Lagrange multiplier to enforce the ns/np condition. The moving force equation is derived directly from the exact ns/np boundary condition, resulting in a smaller, well-conditioned coefficient matrix compared to IBPM. MIBM demonstrated good results and improved performance. However, its increased implementation complexity and higher computational memory requirements, especially with more Lagrangian markers or multiple bodies, make it less appealing.

A different method to enhance the enforcement of the ns/np boundary condition involves refining the mathematical reciprocity of IBM interpolation and spreading operators. This refinement directly influences the calculation of the Lagrangian weight used in the spreading operator. Previous studies [10,16,17] demonstrated that assigning the Lagrangian weight to be equal to the Eulerian weight, as initially proposed by Uhlmann [8], disrupts this reciprocal property, resulting in ns/np boundary errors and fluid intrusion into the solid domain. To address this issue, Pinelli et al. [18] suggested a method for determining the Lagrangian weight by solving an additional set of linear equations, thereby enhancing the reciprocity of the interpolation and spreading processes. Additionally, a different study [17] introduced an analytical correction for the Lagrangian weight that bypasses the need for solving a linear system, with a subsequent improvement [10] to consider the curvature of the immersed boundary.

In the current work, the performance and efficiency of the implicit DF-IBM are improved by speeding up the process to fulfill the ns/np boundary condition through the following points, (i) different Lagrangian weight methods that improve the reciprocity of the IBM-related operators, (ii) additional Lagrangian weight correction, and (iii) Lagrangian boundary force inheritance scheme. The current scheme is coupled with the cell-centered finite volume fluid Navier–Stokes equations through the PISO solution algorithm found in the open-source C++ toolbox OpenFOAM environment. The predictor–corrector nature of the PISO algorithm is used to decouple all fluid variables, pressure, velocity, and boundary force in a segregated fashion. This scheme ensures that the fluid-solid system's dual constraints problem, represented by the divergence-free condition on the Eulerian domain and the ns/np condition on the Lagrangian domain, are exactly fulfilled. In addition, a modified PPE is produced that includes the boundary force term already satisfying the latter condition allowing the usage of existing fast PPE solvers in OpenFOAM without any modifications.

## 2. Direct-forcing immersed boundary method (DF-IBM)

### 2.1. The DF-IBM governing equations

The formulation derived couples the fluid Navier–Stokes equations with the underlying solid Lagrangian domain. The general governing equations of the DF-IBM are derived as follows:

$$\nabla \cdot \mathbf{u} = 0, \tag{1a}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) - \nabla \cdot (\nu \nabla \mathbf{u}) = -\frac{1}{\rho}\nabla p + \mathbf{f}, \tag{1b}$$

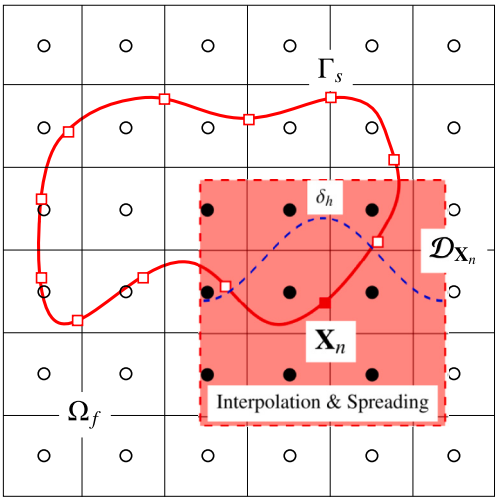



**Fig. 1.** Interpolation and spreading operators with the help of a smoothed approximation of the Dirac delta function in the DF-IBM.

$$\mathbf{U}_{IB}(\mathbf{X}_n, t) = \mathcal{I}[\mathbf{u}(\mathbf{x}_{i,j}, t)], \tag{1c}$$

$$\mathbf{f}(\mathbf{x}_{i,j}, t) = \mathcal{S}[\mathbf{F}_{IB}(\mathbf{X}_n, t)], \tag{1d}$$

where $\nu$ is the effective molecular kinematic viscosity, $\rho$ is the density and $p$ is the pressure. $\mathbf{x}_{i,j}$ is the fluid position of an Eulerian grid point, $\mathbf{X}_n$ is the boundary position of a Lagrangian marker, $\mathbf{u}(\mathbf{x}_{i,j}, t)$ is the fluid velocity of an Eulerian grid point, $\mathbf{U}_{IB}(\mathbf{X}_n, t)$ is the desired solid immersed boundary velocity at each Lagrangian marker, $\mathbf{f}(\mathbf{x}_{i,j}, t)$ is the Eulerian boundary force exerted on the Eulerian grid points, $\mathbf{F}_{IB}(\mathbf{X}_n, t)$ is the singular Lagrangian boundary force at each Lagrangian marker, and $\mathcal{I}$ and $\mathcal{S}$ are two linear operators that manage the data transfer between Eulerian and Lagrangian grids.

### 2.2. IBM-related linear operators

The data transfer between the Eulerian and Lagrangian grids is managed by two separate linear operators using a smoothed approximation of the Dirac delta function $\delta$ as shown in Fig. 1.

(a) Velocity interpolation operator:

$$\mathbf{U}(\mathbf{X}_n, t) = \mathcal{I}[\mathbf{u}(\mathbf{x}_{i,j}, t)] = \sum_{\mathbf{x}_{i,j} \in \mathcal{D}_{\mathbf{X}_n}} \mathbf{u}(\mathbf{x}_{i,j}, t)\delta_h(\mathbf{x}_{i,j} - \mathbf{X}_n)\Delta V_{i,j}. \tag{2}$$

This operator represents the interpolation of the Eulerian velocity field back to each Lagrangian marker $\mathbf{X}_n$. For Cartesian uniform meshes the Eulerian weight $\Delta V_{i,j} = h^2$ (for $2D$ topology), with $h$ being the distance between two Eulerian grid points, and $(\mathbf{x}_{i,j} \in \mathcal{D}_{\mathbf{X}_n})$ represents the collection of all neighboring Eulerian grid points that fall inside the support domain $\mathcal{D}_{\mathbf{X}_n}$ of the Lagrangian marker $\mathbf{X}_n$ shown by the dashed square in Fig. 1.

(b) Force spreading operator:

$$\mathbf{f}(\mathbf{x}_{i,j}, t) = \mathcal{S}[\mathbf{F}_{IB}(\mathbf{X}_n, t)] = \sum_{\mathbf{X}_n \in \mathcal{D}_{\mathbf{x}_{i,j}}} \mathbf{F}_{IB}(\mathbf{X}_n, t)\delta_h(\mathbf{x}_{i,j} - \mathbf{X}_n)W_n. \tag{3}$$

The computed Lagrangian force is spread back to the neighboring Eulerian grid points in the support domain of the Lagrangian marker $\mathbf{X}_n$. The Lagrangian weight $W_n$ is the weight associated to the solid immersed boundary marker, and $(\mathbf{X}_n \in \mathcal{D}_{\mathbf{x}_{i,j}})$ refers to a loop over all the Lagrangian markers whose support domain $\mathcal{D}_{\mathbf{X}_n}$ contains the same Eulerian grid point $\mathbf{x}_{i,j}$.

As for the interpolation function, the Dirac delta function in its discrete form was adopted. For Cartesian uniform meshes, the $2D$

**Table 1**
Properties and moment conditions for all continuous kernel functions $\phi$.

| | Continuous $\forall r$ | Support radius $r_s$ | Zeroth moment | First moment | Second moment | Third moment | Sum of squares condition |
|---|---|---|---|---|---|---|---|
| $\phi_1$ | ✓ | 1.5 | ✓ | ✓ | × | × | ✓($C = 0.5$) |
| $\phi_2$ | ✓ | 2.5 | ✓ | ✓ | ✓($K = \frac{38}{60} - \frac{\sqrt{69}}{60}$) | ✓ | ✓($C = 0.3925$) |

approximation of the discrete Dirac delta function is given by:

$$\delta_h(\mathbf{x}_{i,j} - \mathbf{X}_n) = \frac{1}{h^2}\phi\left(\frac{x_{i,j} - X_n}{h}\right)\phi\left(\frac{y_{i,j} - Y_n}{h}\right), \tag{4}$$

where $\phi$ is a continuous function named kernel function that scales to the mesh size $h$ inside the Lagrangian support domain $\mathcal{D}_{\mathbf{X}_n}$ centered at each Lagrangian marker $\mathbf{X}_n = (X_n, Y_n)$, which encloses all neighboring Eulerian grid points $\mathbf{x}_{i,j} = (x_{i,j}, y_{i,j})$ used for the interpolating and spreading operators. The continuous kernel function $\phi$ can take various forms. In the present work, the 3-point-width function $\phi_1$ proposed in [19] was used as the kernel function, in addition to the 5-point-width function $\phi_2$ developed in [20], since it suppresses the non-physical force oscillations when a moving immersed boundary is simulated due to its high moment condition [21].

Both kernel functions satisfy the zeroth and first moment conditions. While the second and third moment conditions are only satisfied by $\phi_2$. Both kernel functions satisfy the sum of squares condition. All mentioned conditions of the kernel function $\phi$ are summarized in Table 1. A summary of all properties and moment conditions of a kernel function can be found in [20].

In addition, the immersed boundary is represented by a set of Lagrangian markers that are uniformly distributed over the boundary, having the arc length $\Delta s$, which is the distance between two consecutive Lagrangian markers, approximately equal to the Eulerian grid spacing $h$, i.e. $\Delta s/h \approx 1$.

### 2.3. DF-IBM algorithms review

Different DF-IBM algorithms can be found in the literature, implemented within a variety of numerical techniques for solving partial differential equations ranging from Finite Different Method (FDM) [8], Finite Volume Method (FVM) [18], Finite Element Method (FEM) [22], and Lattice Boltzmann Method (LBM) [16]. These DF-IBM algorithms are inserted inside the incompressible Navier–Stokes equations where $p - \mathbf{U}$ coupling is handled by using either the projection method (fractional step method) and its variants as in [15,18], the SIMPLE algorithm [23], the PISO algorithm [24,25]. Regardless, of the numerical technique and $p - \mathbf{U}$ decoupling algorithm used, the DF-IBM steps present in every approach are always identical. The generic steps are ordered as follows:

1. Eulerian velocity interpolation:

$$\mathbf{U}(\mathbf{X}_n, t) = \mathcal{I}[\mathbf{u}^*(\mathbf{x}_{i,j}, t)]. \tag{5}$$

   Here we interpolate a predicted Eulerian velocity field $\mathbf{u}^*$, that ignores the presence of a solid immersed boundary inside the fluid region, to obtain the Lagrangian velocity $\mathbf{U}$ on the Lagrangian marker $\mathbf{X}_n$.
2. Direct-forcing:

$$\mathbf{F}_{IB}(\mathbf{X}_n, t) = \frac{\mathbf{U}_{IB}(\mathbf{X}_n, t) - \mathbf{U}(\mathbf{X}_n, t)}{\Delta t}, \tag{6}$$

   where $\mathbf{U}_{IB}$ represents the desired Lagrangian velocity of the immersed boundary and is directly imposed at the Lagrangian marker $\mathbf{X}_n$, which can be determined either using a prescribed equation of motion or from the coupling with a solid structural solver. For stationary immersed bodies $\mathbf{U}_{IB}$ is set equal to zero.
3. Lagrangian force spreading:

$$\mathbf{f}(\mathbf{x}_{i,j}, t) = \mathcal{S}[\mathbf{F}_{IB}(\mathbf{X}_n, t)]. \tag{7}$$

   The Lagrangian boundary force $\mathbf{F}_{IB}$ is spread from each Lagrangian marker $\mathbf{X}_n$ onto their corresponding neighboring Eulerian grid points to obtain the Eulerian boundary force $\mathbf{f}$.
4. Eulerian velocity update:
   This step is used to update and correct the predicted velocity field $\tilde{\mathbf{u}}$ to reckon the presence of the solid immersed boundary inside the fluid region. The equation used to update the Eulerian velocity is not generic, but solver dependent. The explicit correction using the implicit Euler scheme (also known as the Backward Euler method) for time discretization yields the following correction:

$$\tilde{\mathbf{u}}(\mathbf{x}_{i,j}, t) = \mathbf{u}^*(\mathbf{x}_{i,j}, t) + \Delta t\mathbf{f}(\mathbf{x}_{i,j}, t). \tag{8}$$

### 2.4. Immersed boundary ns/np condition

The spreading and the interpolation operators in the DF-IBM are reciprocal, i.e. inversely related. For instance, if we interpolate a vector field $\mathbf{u}$ from the Eulerian to the Lagrangian domain, the resulting vector in the Lagrangian domain is $\mathcal{I}[\mathbf{u}]$. Then we spread the obtained vector back to the Eulerian domain, i.e., $\mathcal{S}[\mathcal{I}[\mathbf{u}]]$, we obtain the same vector field $\mathbf{u}$. Once this reciprocity is violated, a ns/np boundary error is generated, allowing the fluid to penetrate the solid domain.

The ns/np boundary error disappears once $\mathbf{U}_{IB}(\mathbf{X}, t) = \mathcal{I}[\tilde{\mathbf{u}}]$, i.e. the ns/np boundary condition is exactly enforced on the Lagrangian Markers $\mathbf{X}$. Employing the concept of multiple DF-IBM [9] helps in satisfying the mentioned equality and improves the reciprocity on the interpolation and spreading operations as demonstrated in [26].

Additionally, the ns/np error can be directly related to the Lagrangian weight $W_n$, used in the force spreading operator. According to Uhlmann [8], the Lagrangian weight was initially set equal to the Eulerian weight. However, this method was found to be inaccurate [10] because the Lagrangian representation of the immersed boundary, described only through its boundary rather than its entire volume, has a dimensional issue, resulting in a missing dimension in the Lagrangian weight used in the force spreading operator. Hence, the Eulerian and Lagrangian weights belong to two distinct discretization spaces as mentioned in [10], and therefore should not be equal. Different methods to calibrate the Lagrangian weight computations can improve the mathematical reciprocity of the interpolation/spreading operators. The methods used in this research are described in the next section.

Besides the DF-IBM iterations, another approach to reduce the ns/np boundary error is by decreasing the simulation time-step, as reported in [11,12,17].

#### 2.4.1. Lagrangian weight computation methods

The Lagrangian weight can be primarily set equal to the Eulerian weight. This approach was adopted by most researchers [8,11,15,27]:

$$W_n = \Delta V_{i,j}. \tag{9}$$

Pinelli et al. [18] proposed to enforce the reciprocity on the interpolation and spreading operators, where the Lagrangian weight is computed by solving the following system of linear equations:

$$\mathbf{A}\mathbf{W}_n = \mathbf{1}, \tag{10}$$

where the vectors $\mathbf{W}_n = [W_1, \ldots, W_{N_{Lag}}]^{\mathrm{T}}$ and $\mathbf{1} = [1, \ldots, 1]^{\mathrm{T}}$ have the dimension of $N_{Lag}$, with $N_{Lag}$ being the total number of Lagrangian markers. $\mathbf{A}$ is the matrix defined by:

$$\mathbf{A} = A_{m,n} = \sum_{\mathbf{x}_{i,j} \in \mathcal{D}_{\mathbf{X}_m}} \delta_h(\mathbf{x}_{i,j} - \mathbf{X}_m)\delta_h(\mathbf{x}_{i,j} - \mathbf{X}_n)h^2.$$

The system is solved efficiently by using the biconjugate gradient stabilized (BiCGStab) Krylov subspace solver. This method is advantageous because it can be applied to both uniform and non-uniform Eulerian grids, and it reduces the ns/np boundary error [10]. However, it increases the computational cost compared to the previous method due to the iterative solution of the system of linear equations, with the computational cost being directly proportional to the total number of Lagrangian markers $N_{Lag}$.

Furthermore, an analytical correction for the Lagrangian weight was proposed in [16,17], based on two main assumptions. Firstly, the curvature of the immersed boundary is significantly small so the local distance can be represented by a straight wall. Secondly, the variation of the Lagrangian force along the immersed boundary is considered small, which means that no discontinuities are taken into consideration. The correction is simply the product of the Lagrangian weight with the inverse of the constant $C$ obtained from the sum of squares condition imposed by the kernel function used, as shown in Table 1. This means that this correction is kernel function dependent since the sum of squares condition is not satisfied by all existing kernel functions. The described correction was shown in [16,17] to eliminate some of the ns/np error generated by the non-reciprocity of the DF-IBM linear operators. The aforementioned correction was later improved to include the curvature of the immersed boundary by Cheylan et al. [10]:

$$W_n = \frac{1}{\sum_{\mathbf{x}_{i,j} \in \mathcal{D}_{\mathbf{X}_m}} \sum_{\mathbf{X}_n \in \mathcal{D}_{\mathbf{x}_{i,j}}} \delta_h(\mathbf{x}_{i,j} - \mathbf{X}_n)\delta_h(\mathbf{x}_{i,j} - \mathbf{X}_m)h^2}. \tag{11}$$

The first sum is performed on each Lagrangian marker $\mathbf{X}_n$ whose support contains the same Eulerian grid points $\mathbf{x}_{i,j}$, while the second sum is performed on all Eulerian grid points $\mathbf{x}_{i,j}$ that fall inside the support domain of $\mathbf{X}_m$. This method can be used independently of the selected kernel function and its computational cost is negligible compared to the method developed by Pinelli et al. [18]. The difference between Eqs. (10)–(11) for the ns/np boundary error was shown to be minimal in [10].

## 3. Fluid Navier–Stokes equations solver

The fluid solver is governed by the dimensional, incompressible, viscous Navier–Stokes equations, expressed in their integral form as:

$$\int_{V_P} \frac{\partial \mathbf{u}}{\partial t}\, dV + \int_{V_P} \nabla\cdot(\mathbf{u}\mathbf{u})\, dV - \int_{V_P} \nabla\cdot(\nu\nabla\mathbf{u})\, dV = -\int_{V_P} \nabla p\, dV + \int_{V_P} \mathbf{f}\, dV. \tag{12}$$

### 3.1. Navier–Stokes finite volume linear algebraic discretized form

OpenFOAM is based on the finite volume space discretization. The Gauss theorem is used to reduce volume integrals to surface integrals, and then using the midpoint rule, the surface integrals are transformed into a discrete sum over all faces of the control volume. This yields the below algebraic equation:

$$\left(\frac{\partial \mathbf{u}_P}{\partial t}\right)^{n+1} + \frac{1}{V_P}\sum_f \phi_f^n \mathbf{u}_f^{n+1} - \frac{1}{V_P}\sum_f \nu_f (\nabla\mathbf{u})_f^{n+1}\cdot\mathbf{S}_f = -(\nabla p)_P^n + \mathbf{f}_P^{n+1}, \tag{13}$$

where $\mathbf{S}_f$ is the face area normal vector between the current control volumes $P$ and $N$, as illustrated in Fig. 2 for Cartesian uniform meshes, equal to $\mathbf{S}_f = A_f\mathbf{n}_f$, with $A_f$ the surface area of the face $f$ and $\mathbf{n}_f$ the normal vector to the face $f$. $V_P$ is the control volume of $P$.

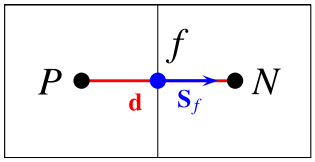



**Fig. 2.** Face area normal vector $\mathbf{S}_f$ for Cartesian uniform meshes.

The temporal term is discretized using a fully implicit three-level second-order scheme, also called the BDF2 scheme (Backward Differentiation Formula of order 2). The face-center values of all explicit quantities $\phi_f^n$ are computed via linear interpolation, also called the central differencing (CD) interpolation scheme. For the convection term, the implicit face variable $\mathbf{u}_f^{n+1}$, is discretized using the NVD convection interpolation scheme, named the Gamma differencing scheme [28]. For the diffusion term, due to mesh uniformity and constant molecular viscosity, the face viscosity $\nu_f$ is the same as the laminar molecular kinematic viscosity $\nu$. The face gradient term $(\nabla\mathbf{u})_f^{n+1}\cdot\mathbf{S}_f$ is calculated for a purely orthogonal mesh using a central difference approximation of the first-order derivative in the direction of the vector $\mathbf{d}$ from the two values of the owner cell $P$ and neighboring cell $N$ around the face $f$. The cell-centered pressure gradient term is computed explicitly in the predictor step of the PISO solution algorithm using the Gauss theorem, having the face values obtained from linear interpolation. Another pressure gradient term will appear in the corrector step of the PISO solution algorithm inside the PPE where it will be implicitly treated similarly to the diffusion term.

For the sake of simplicity, the linear algebraic equation obtained has the following form:

$$a_P\mathbf{u}_P^{n+1} + \sum_N a_N\mathbf{u}_N^{n+1} = \mathbf{RHS} - (\nabla p)_P^n + \mathbf{f}_P^{n+1}, \tag{14}$$

where $a_P$ is the diagonal matrix coefficient of owner cell $P$ that arises from the implicit contribution of the temporal, convection, and diffusion terms discretization, $a_N$ is the off-diagonal matrix coefficient of neighboring cell $N$ resulting from the implicit contribution of the convection and diffusion terms, and **RHS** represents the source term accounting for the explicit contribution of the transient term.

### 3.2. Modified PISO solution algorithm

The pressure and velocity fields in an unsteady incompressible flow are coupled through the continuity equation. Hence, solving the fluid system necessitates the simultaneous determination of both fields in order to ensure a divergence-free velocity.

The present work adopts the segregated Pressure Implicit with Splitting of Operators (PISO) algorithm [29] for pressure–velocity decoupling. The built-in solver is modified to incorporate the solid immersed boundary using DF-IBM. The PISO algorithm is based on the predictor–corrector procedure where it features an implicit momentum predictor step at the beginning of the time-step, treating an estimated pressure field as known, followed by multiple explicit momentum corrector steps (usually 2 to 3) to enforce a divergence-free velocity field through the pressure equation using the projection method.

The implicit predictor step is given by Eq. (14) excluding the boundary force term and using the pressure solution from the previous time level. The equation for the predicted velocity field is reformulated as follows:

$$a_P\mathbf{u}_P^* + \sum_N a_N\mathbf{u}_N^* = \mathbf{RHS} - (\nabla p)_P^n, \tag{15}$$

where $\mathbf{u}^*$ is the predicted velocity field.

The multiple explicit corrector steps are derived as follows. Referring back to Eq. (14), the neighboring velocities term is moved to the

right-hand side of the equation, while the pressure gradient term is now treated implicitly as an unknown:

$$a_P \mathbf{u}_P^{n+1} = \mathbf{H}(\mathbf{u}_N^{n+1}) - (\nabla p)_P^{n+1} + \mathbf{f}_P^{n+1}, \tag{16}$$

where the linear operator $\mathbf{H}$ is described by:

$$\mathbf{H}(\mathbf{u}_N^{n+1}) = -\sum_N a_N \mathbf{u}_N^{n+1} + \mathbf{RHS}. \tag{17}$$

The implicit velocity in the $\mathbf{H}$ operator is replaced by the predicted velocity field $\mathbf{u}^*$ to produce the explicit behavior of the corrector step. This formulation allows for the definition of a new cell-centered velocity:

$$\mathbf{u}_P^{n+1} = \frac{\mathbf{H}(\mathbf{u}_N^*)}{a_P} - \frac{1}{a_P}(\nabla p)_P^{n+1} + \frac{1}{a_P}\mathbf{f}_P^{n+1}. \tag{18}$$

Taking the divergence of Eq. (18) and enforcing the divergence-free condition at the new time level results in the continuity equation, also known as the divergence-free condition:

$$\nabla \cdot \mathbf{u}_P^{n+1} = 0. \tag{19}$$

To derive the discrete pressure equation, known as the PPE, the previous velocity equation Eq. (18) is substituted into the continuity equation Eq. (19):

$$\nabla \cdot \left(\frac{1}{a_P}(\nabla p)_P^{n+1}\right) = \nabla \cdot \left(\frac{\mathbf{H}(\mathbf{u}_N^*)}{a_P}\right) + \nabla \cdot \left(\frac{1}{a_P}\mathbf{f}_P^{n+1}\right). \tag{20}$$

By solving the modified PPE (mPPE) given by Eq. (20), the pressure solution $p^{n+1}$ is obtained. This solution includes both the physical pressure field, derived from the flow field and a correction component that enforces mass continuity and compensates for errors in the initial pressure field.

Finally, the cell-centered velocity is explicitly corrected to also satisfy the divergence-free condition:

$$\mathbf{u}_P^{n+1} = \frac{\mathbf{H}(\mathbf{u}_N^*)}{a_P} - \frac{1}{a_P}(\nabla p)_P^{n+1} + \frac{1}{a_P}\mathbf{f}_P^{n+1}. \tag{21}$$

It is worth noting that the pressure equation, referred to as the mPPE, is the primary bottleneck in the numerical simulation process. Solving the pressure equation is the most time-consuming part of the simulation due to its high condition number. In this work, the mPPE results in a symmetrical coefficient matrix. It was found that the Conjugate Gradient (CG) iterative solver, combined with the Geometric Agglomerated Algebraic Multigrid (GAMG) preconditioner and the Gauss–Seidel smoother, is more computationally efficient than all other solvers and preconditioners combinations.

## 4. Proposed FIDF-IBM algorithm

In this work, we propose an efficient fast implicit DF-IBM (FIDF-IBM) that reduces the computational effort compared to previous implicit IBM, such as the IBPM of Taira and Colonius [13] and the IBM algorithm of Ji et al. [14]. This variant addresses the fluid-solid system's dual constraints problem for an incompressible flow problem sequentially using the PISO algorithm as the Navier–Stokes fluid solver.

Modifications to the original PISO algorithm are presented here to ensure the velocity field satisfies two constraints within the predictor–corrector algorithm:

(a) Ns/np velocity boundary condition on the Lagrangian markers, ${\mathbf{U}_{IB}}^{n+1} = \mathcal{I}[\tilde{\mathbf{u}}]$ on $\Gamma_s$,
(b) Divergence-free condition on Eulerian grid points, $\nabla \cdot \mathbf{u}^{n+1} = 0$ in $\Omega_f$.

Additionally, a stability criterion for the PISO algorithm must be met, requiring the Courant (CFL) number to be less than one. The two constraints are decoupled from the Navier–Stokes momentum equations through the operator splitting strategy of the PISO algorithm. The following algorithm is adopted at each time-step, solving for $\mathbf{u}$, $p$, and $\mathbf{f}$ sequentially until convergence is achieved, where the superscripts $n$ and $n+1$ denote the previous (known) and current (unknown) time levels, respectively:

1. Momentum predictor, without the addition of the boundary force term and having the pressure taken from previous time-step $t^n$:

$$a_P \mathbf{u}_P^* + \sum_N a_N \mathbf{u}_N^* = \mathbf{RHS} - (\nabla p)^n. \tag{22}$$

Note that $\nabla \cdot \mathbf{u}^* \neq 0$.

2. DF-IBM Iterations procedure, with inner iterations start for $m = 1$, having $\tilde{\mathbf{u}}^{(0)} = \mathbf{u}^*$ and ${\mathbf{F}_{IB}}^{(0)} = {\mathbf{F}_{IB}}^n$:

(a) Eulerian velocity interpolation:

$$\mathbf{U}^{(m-1)} = \mathcal{I}[\tilde{\mathbf{u}}^{(m-1)}]. \tag{23}$$

(b) Direct-forcing:

$${\mathbf{F}_{IB}}^{(m)} = \frac{{\mathbf{U}_{IB}}^{n+1} - \mathbf{U}^{(m-1)}}{\Delta t} + {\mathbf{F}_{IB}}^{(m-1)}. \tag{24}$$

(c) Lagrangian force spreading:

$$\mathbf{f}^{(m)} = \mathcal{S}[{\mathbf{F}_{IB}}^{(m)}]. \tag{25}$$

(d) Eulerian velocity update:

$$a_P \tilde{\mathbf{u}}_P^{(m)} + \sum_N a_N \tilde{\mathbf{u}}_N^{(m)} = \mathbf{RHS} - (\nabla p)^n + \mathbf{f}^{(m)}. \tag{26}$$

(e) The ns/np velocity boundary condition requirement:

$${\mathbf{U}_{IB}}^{n+1} = \mathcal{I}[\tilde{\mathbf{u}}^{(m)}] = \mathbf{U}^{(m)}. \tag{27}$$

A convergence criterion is set for two consecutive DF-IBM iterations:

$$\|E_{ns/np}\|_2 = \|E_{\tilde{\mathbf{u}}^{(m)}}\|_2 - \|E_{\tilde{\mathbf{u}}^{(m-1)}}\|_2 < \epsilon, \tag{28}$$

with

$$\|E_{\tilde{\mathbf{u}}^{(m)}}\|_2 = \|{\mathbf{U}_{IB}}^{n+1} - \mathbf{U}^{(m)}\|_2,$$

and

$$\|E_{\tilde{\mathbf{u}}^{(m-1)}}\|_2 = \|{\mathbf{U}_{IB}}^{n+1} - \mathbf{U}^{(m-1)}\|_2,$$

where $\epsilon$ is a user-specified tolerance, based on whether an accurate ns/np boundary condition is required or not. The $L_2$ norm is computed using Eq. (A.1) provided in the appendix section.

(f) If the defined convergence criterion is reached in Step 2 (e), then:

$$\mathbf{f}^{n+1} = \mathbf{f}^{(m)} \quad \& \quad \tilde{\mathbf{u}} = \tilde{\mathbf{u}}^{(m)}, \tag{29}$$

otherwise, return to Step 2 (a), and let $m = m+1$. It can be noted at this stage that $\nabla \cdot \tilde{\mathbf{u}} \neq 0$, while the ns/np boundary condition is fulfilled $\mathbf{U}^{(m)} = \mathcal{I}[\tilde{\mathbf{u}}^{(m)}] = \mathbf{U}_{IB}$.

3. PISO corrector loops, with inner loops start for $m = 1$, having $1 \leq m \leq m_f$ and $m_f = 3$. For $m = 1$, $\hat{\mathbf{u}}^{(0)} = \tilde{\mathbf{u}}$:

(a) The divergence-free constraint is enforced by using the mPPE represented by Eq. (20) to compute the new pressure field:

$$\nabla \cdot \left(\frac{1}{a_P}(\nabla p)^{(m)}\right) = \nabla \cdot \left(\frac{\mathbf{H}(\hat{\mathbf{u}}^{(m-1)})}{a_P}\right) + \nabla \cdot \left(\frac{1}{a_P}\mathbf{f}^{n+1}\right). \tag{30}$$

(b) Afterward, the velocity field is corrected explicitly from the former pressure solution $p^{(m)}$ using Eq. (21):

$$\hat{\mathbf{u}}^{(m)} = \frac{\mathbf{H}(\hat{\mathbf{u}}^{(m-1)})}{a_P} - \frac{1}{a_P}(\nabla p)^{(m)} + \frac{1}{a_P}\mathbf{f}^{n+1}. \tag{31}$$

(c) At $m = m_f$, the velocity and pressure fields are updated at time-step $t^{n+1}$:

$$\mathbf{u}^{n+1} = \hat{\mathbf{u}}^{(m_f)}, \tag{32}$$

$$p^{n+1} = p^{(m_f)}. \tag{33}$$

Note that $\nabla \cdot \mathbf{u}^{n+1} = 0$.

**Algorithm 1:** FIDF-IBM algorithm

```
Given p^n and u^n;
Predict initial Eulerian velocity u* using Eq. (22);
Initialize: m = 1, ũ^(0) = u* and F_IB^(0) = F_IB^n;
while m ≤ m_max do
    Interpolate Eulerian velocity to obtain U^(m−1) using Eq. (23);
    Compute singular Lagrangian force F_IB^(m) using Eq. (24);
    Spread Lagrangian force to obtain f^(m) using Eq. (25);
    Update the Eulerian predicted velocity ũ^(m) using Eq. (26);
    if ‖E_ns/np‖_2 < ϵ then
        f^(n+1) = f^(m) and ũ = ũ^(m);
        break;
    else
        m = m + 1;
    end
end
Initialise: m_f = 3 and û^(0) = ũ;
for m = 1 to m_f do
    Solve mPPE for p^(m) using Eq. (30);
    Correct Eulerian velocity û^(m) using Eq. (31);
    if m = m_f then
        u^(n+1) = û^(m_f) and p^(n+1) = p^(m_f);
    end
end
```

The FIDF-IBM algorithm is summarized in Algorithm 1. Based on the original PISO method of Issa [29], the proposed algorithm enforces the ns/np boundary condition on the Lagrangian domain and the divergence-free condition on the Eulerian domain in a segregated manner. Albeit the former condition is satisfied before the PISO algorithm's corrector loops to enforce the divergence-free condition, the Lagrangian interpolated velocity remains largely unchanged even if computed after the corrector loops. This is consistent with previous findings [11,15]. Similarly to previous works [14], our modified PPE differs from the PPE used in IPBM [13] by including the boundary force that already satisfies the ns/np boundary condition. Conversely to the IBPM [13], the divergence-free constraint is only fulfilled by the mPPE. This avoids changing the coefficient matrix and undermining the condition number of the PPE, which leads to convergence problems as in the case of the IBPM, allowing the use of fast iterative solvers such as the GAMG solver or the Krylov subspace CG solver without any prior modifications.

An additional attractive aspect of this algorithm is the accelerated process to achieve the ns/np boundary condition on the Lagrangian domain through three simple refinements. Firstly, by replacing Eq. (9) of Uhlmann [8] for computing the Lagrangian weight, with Eq. (10) of Pinelli et al. [18] or Eq. (11) from [10], where the reciprocity of the IBM-related operators is improved. The current authors recommend using Eq. (11) for moving immersed boundaries, since for moving boundaries, the Lagrangian weight is re-computed at each time level, therefore avoiding the increase of the computational effort of the algorithm by solving the linear system shown in Eq. (10). In this case, the analytical Lagrangian weight computation found in [10] is recommended. Secondly, it was established through trial and error, that an additional Lagrangian weight correction of 3/2 to Eqs. (10) and (11) can be applied for the Lagrangian weight computation. Multiplying the Lagrangian weight by this factor notably speeds up the ns/np boundary convergence while keeping the forces on the immersed boundary intact. Finally, the inheritance scheme for the Lagrangian boundary force $\mathbf{F}_{IB}{}^{(m-1)}$ used in Eq. (24), also called the initialization method, was largely affecting the efficiency of the ns/np boundary convergence. It is proposed here to inherit the values of $\mathbf{F}_{IB}{}^{(0)}$ for $m = 1$ at the beginning of a new time level by the value from the previous time level, i.e. $\mathbf{F}_{IB}{}^{(0)} = \mathbf{F}_{IB}{}^{n}$, instead of initializing the value by zero.

A convergence criterion for the ns/np boundary condition was established for two consecutive DF-IBM iterations. As reported by [9], the error convergence rate significantly decreases after several iterations, which means that additional iterations will not impact the desired solution. This considerably reduces the number of iterations per time-step.

## 5. Results and discussions

### 5.1. FIDF-IBM performance study

The performance of different DF-IBM strategies is assessed quantitatively and qualitatively against the present FIDF-IBM, where the PISO decoupling algorithm is dropped out for this study. The performance is analyzed through the following test:

(a) Initialize velocity field in $x$-direction:

$$u_0(x, y) = x^2 - y^2 - 1 \quad \text{in } \Omega_f,$$

(b) Find $f$ such that:

$$U_{IB}(X, Y) = \mathcal{I}[\tilde{u}^{(m)}(x, y)] = \mathcal{I}[u_0(x, y) + \Delta t f^{(m)}] \quad \text{on } \Gamma_s.$$

The following study involves a circular cylinder with a diameter of 0.4, centered at $(0.5, 0.5)$ within an Eulerian square domain $\Omega_f = [0, 1] \times [0, 1]$. The cylinder is stationary, thus $U_{IB} = 0$. The domain is discretized using $200 \times 200$ Eulerian grid points. The cylinder boundary $\Gamma_s$ is represented by 248 Lagrangian markers. The time-step $\Delta t$ is set to 1 to represent a single iteration, and the kernel function $\phi_1$ is utilized. Slip boundary conditions are enforced on all boundaries of the domain.

For this specific analysis, a stricter convergence criterion is established, differing from the one outlined in Eq. (28) for two consecutive DF-IBM iterations, as follows:

$$\begin{aligned} \|E_{ns/np}\|_2 &= \|E_{\tilde{u}^{(m)}}\|_2 \\ &= \|U_{IB}(X_n, Y_n) - \mathcal{I}[\tilde{u}^{(m)}(x, y)]\|_2 < \epsilon, \end{aligned} \tag{34}$$

since we are interested in the updated ns/np boundary error through the newly updated velocity field and not the error between two successive DF-IBM iterations.

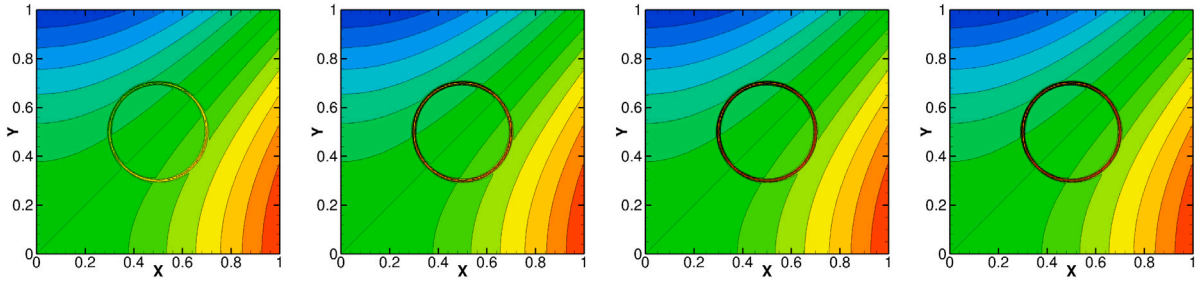
Table 2 presents a comparison of the number of iterations required to achieve the ns/np boundary constraint, the CPU time taken to meet this constraint, and the boundary error measured using the $L_2$ norm. The tolerance was set to $1 \times 10^{-15}$ for the implicit DF-IBM strategies, using the convergence criterion defined by Eq. (34).

The explicit DF-IBM of Uhlmann [8], Pinelli et al. [18], and Cheylan et al. [10] all required nearly the same CPU time since only one iteration is needed due to their explicit nature, excluding the time needed to compute the Lagrangian weight. The Lagrangian weight method using Eqs. (10) and (11) reduced the ns/np error by a factor of 6800 and 113 respectively, when compared to Uhlmann's result. The multi-DF-IBM methods of Luo et al. [9], Kempe & Fröhlich [11], and Breugem [12] use 3 iterations in their algorithms with the Lagrangian weight method used by Uhlmann [8], i.e., Eq. (9). An error reduction by a factor of 4 was observed. Nevertheless, these errors obtained using the explicit approach are still considered large.

The implicit strategy proposed by Ji et al. [14] required 2853 iterations to reduce the ns/np error to the prescribed tolerance. Conversely, the present FIDF-IBM required 692 and 932 iterations when

**Table 2**
Performance comparison of different DF-IBM variants in terms of iterations, CPU time, and ns/np boundary error.

| DF-IBM variant | Explicit/Implicit strategy | Lagrangian Weight method | Iterations (m) | CPU time (s) | $\|E_{ns/np}\|_2$ |
|---|---|---|---|---|---|
| Uhlmann [8] | Explicit | Eq. (9) | 1 | $4.13\times10^{-3}$ | $5.12\times10^{-1}$ |
| Pinelli et al. [18] | Explicit | Eq. (10) | 1 | $4.14\times10^{-3}$ | $7.53\times10^{-5}$ |
| Cheylan et al. [10] | Explicit | Eq. (11) | 1 | $4.34\times10^{-3}$ | $4.53\times10^{-3}$ |
| Multi-DF-IBM [9,11,12] | Explicit | Eq. (9) | 3 | $6.19\times10^{-3}$ | $1.29\times10^{-1}$ |
| Ji et al. [14] | Implicit | Eq. (9) | 2853 | $1.54\times10^{0}$ | $9.97\times10^{-16}$ |
| Present FIDF-IBM | Implicit | Eq. (10) + Correction | 692 | $4.10\times10^{-1}$ | $9.97\times10^{-16}$ |
| Present FIDF-IBM | Implicit | Eq. (11) + Correction | 932 | $5.44\times10^{-1}$ | $9.87\times10^{-16}$ |




(a) First explicit DF-IBM [8] (b) Multi-DF-IBM [9, 11, 12] (c) Explicit DF-IBM with improved reciprocity [10, 18] (d) Implicit DF-IBM ([14] and the present FIDF-IBM)

**Fig. 3.** Velocity ($u$) contours for different DF-IBM strategies. Where the vorticity contours are varied from −2 (blue) to 0 (red).

using Eqs. (10) and (11) respectively, with the additional Lagrangian weight correction. However, a reduction of nearly 70% in CPU time was achieved with the present FIDF-IBM when compared to the implicit variant of Ji et al. [14].

These results are qualitatively reflected in the velocity contours over the Eulerian domain shown in Fig. 3. Uhlmann's method [8] does not accurately impose the desired Lagrangian velocity on the immersed boundary ($U_{IB} = 0$), represented by the red color in Fig. 3(a). An improvement is noted for the multi-DF-IBM variant used in [9,11,12] as shown in Fig. 3(b). Finally, the results of Pinelli et al. [18] and Cheylan et al. [10] depicted in Fig. 3(c) are nearly identical and indistinguishable from implicit DF-IBM methods shown in Fig. 3(d) from a qualitative perspective, contrary to the quantitative analysis in Table 2.

The remaining investigation demonstrates the proposed inheritance scheme for the Lagrangian boundary force. For this study, the present FIDF-IBM is tested inside the PISO algorithm for transient flows. A circular cylinder with diameter $D = 1$ located at the center of an Eulerian squared domain $\Omega_f = [-2.5, 2.5]\times[-2.5, 2.5]$ is considered. The domain is covered by 250 × 250 Eulerian grid points, with $h = 0.02$. The cylinder is also stationary, i.e., $\mathbf{U}_{IB}$ is set to 0 in all directions and is formed by 156 Lagrangian markers uniformly distributed over $\Gamma_s$. The Reynolds number $Re = u_\infty D/\nu$ is set to 100. The time-step is set to $\Delta t = 0.004$. The time is advanced until the non-dimensional time $tu_\infty/D$ reaches a value of 10. The tolerance for Eq. (28) for two successive DF-IBM iterations is set to $1\times10^{-8}$ and $\phi_1$ was selected as the kernel function.

The Lagrangian force $\mathbf{F}_{IB}{}^{(m-1)}$ found in Eq. (24) of the FIDF-IBM algorithm should be initialized at the start of the iterations, i.e., $\mathbf{F}_{IB}{}^{(0)}$ for $m = 1$. This initialization process can be handled using two different approaches. The first one is by setting the variable $\mathbf{F}_{IB}{}^{(0)} = 0$ at the beginning of the DF-IBM iterations when $m = 1$, meaning that this variable acts as a constant throughout the whole simulation run time. The other approach is achieved when the variable at the current time level inherits its value from the previous time level during the simulation run time, $\mathbf{F}_{IB}{}^{(0)} = \mathbf{F}_{IB}{}^{n}$, ergo the value of this variable is altered at every time-step.

Fig. 4 shows the number of iterations to reach the tolerance for two successive DF-IBM iterations along with the ns/np boundary error in terms of $L_2$ norm, as computed in Eq. (34) against the non-dimensional time. It appears clearly that the proposed initialization process speeds up the calculations to reach the prescribed tolerance, by reducing the number of DF-IBM iterations at $tu_\infty/D = 10$, from 414 to 108 when the inheritance scheme is used. In addition, the ns/np error is reduced by one order of magnitude from $3.81\times10^{-6}$ for the first case to $4.74\times10^{-7}$ for the proposed initialization. Moreover, an asymptotic behavior can be observed for both the number of iterations and the ns/np boundary error against the non-dimensional time.

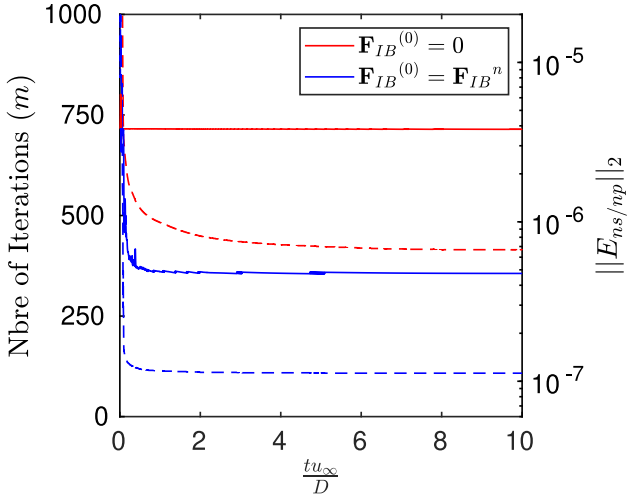
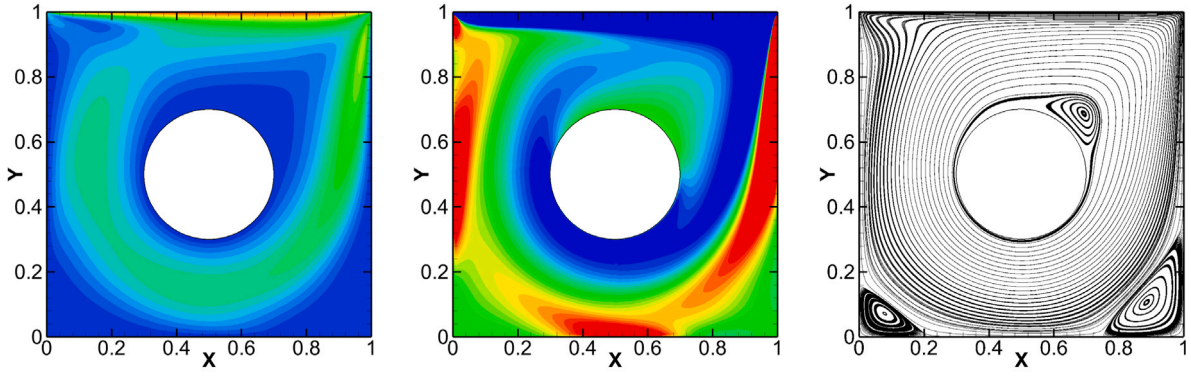



**Fig. 4.** Lagrangian boundary force inheritance scheme on the ns/np condition. Solid lines correspond to the ns/np boundary error and dashed lines correspond to the number of iterations.

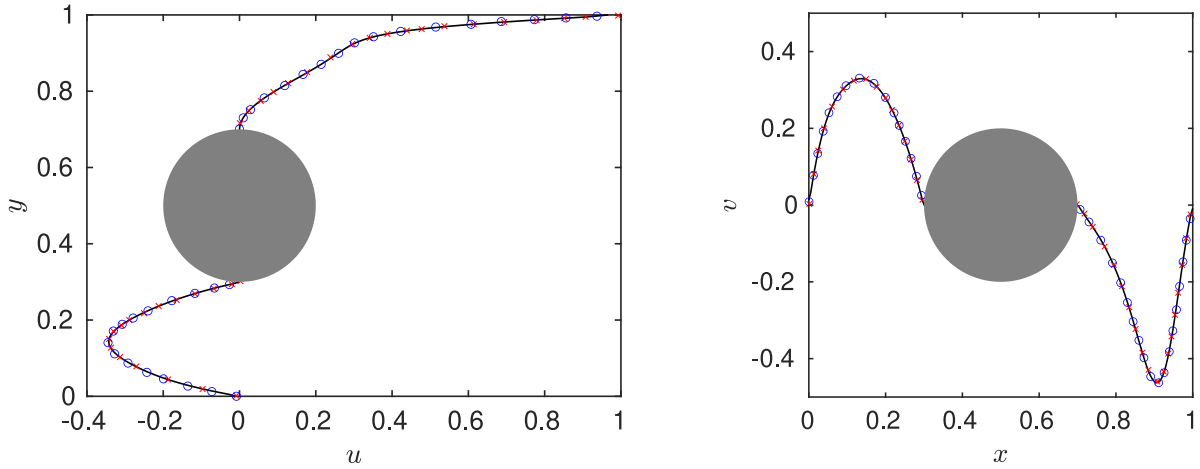
### 5.2. Lid-driven cavity with an embedded cylinder

The flow developed inside a square lid-driven cavity with an embedded cylinder placed at the center is considered. The cavity length



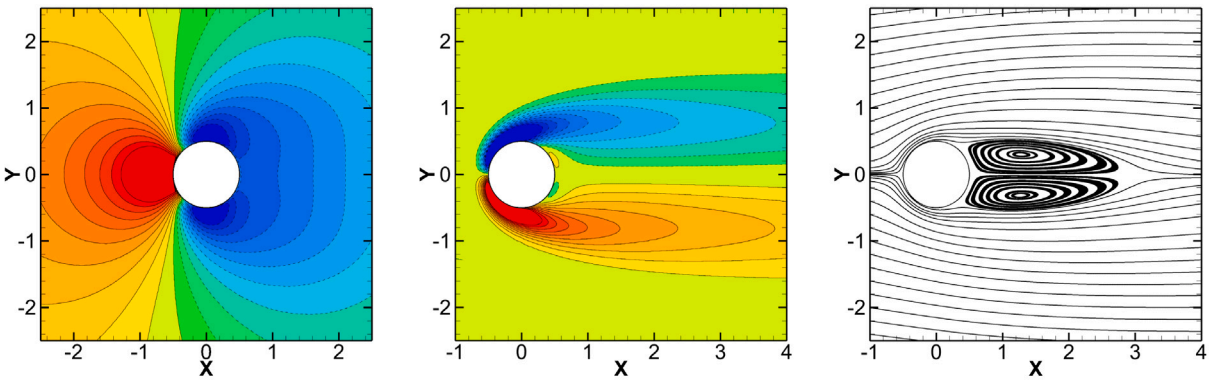

(a) (b) (c)

**Fig. 5.** Velocity magnitude contours (a), vorticity contours (b), and streamlines (c) for the lid-driven cavity with the embedded cylinder at $Re = 1000$. Where the velocity magnitude contours are varied from 0 (blue) to 1 (red) and the vorticity contours are varied from −3 (blue) to 3 (red).




(a) (b)

**Fig. 6.** Velocity $u$ component along $x = 0.5$ (a) and velocity $v$ components along $y = 0.5$ (b) for the lid-driven cavity with the embedded cylinder at $Re = 1000$. Solid lines (——) correspond to the present results, (○) symbols correspond to the data reported in [23], and (×) symbols correspond to the data reported in [15].




(a) (b) (c)

**Fig. 7.** Instantaneous pressure contours (a), vorticity contours (b), and streamlines (c) for the flow over a stationary cylinder at $Re = 40$. Where the pressure contours are varied from −0.3 (blue) to 0.3 (red) and the vorticity contours are varied from −3 (blue) to 3 (red).

is taken as $L = 1$ and the cylinder diameter is set equal to $0.4L$. The boundary conditions correspond to the classical lid-driven cavity problem, with a Dirichlet boundary condition specified at the top ($u_\infty = 1, v_\infty = 0$) and a wall (no-slip) boundary condition specified for the remaining sides ($u = 0, v = 0$). The Reynolds number $Re = u_\infty L/\nu$ based on the cavity length is set to 1000. A steady-state solution governs the flow as the simulation time is advanced. A uniform Eulerian mesh covered by $256 \times 256$ with $h = 3.90625 \times 10^{-3}$ is used similar to the work of [23]. The cylinder being stationary is formed by 320 Lagrangian markers.

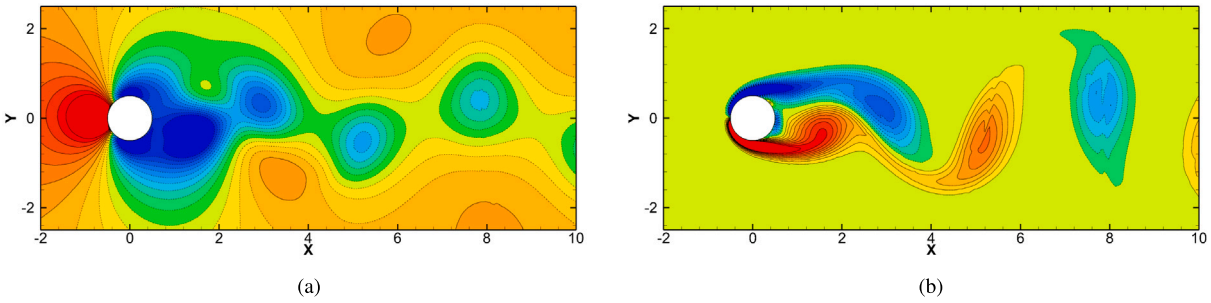



**Fig. 8.** Instantaneous pressure contours (a) and vorticity (b) contours for the flow over a stationary cylinder at $Re = 100$. Where the pressure contours are varied from −0.4 (blue) to 0.3 (red) and the vorticity contours are varied from −3 (blue) to 3 (red).

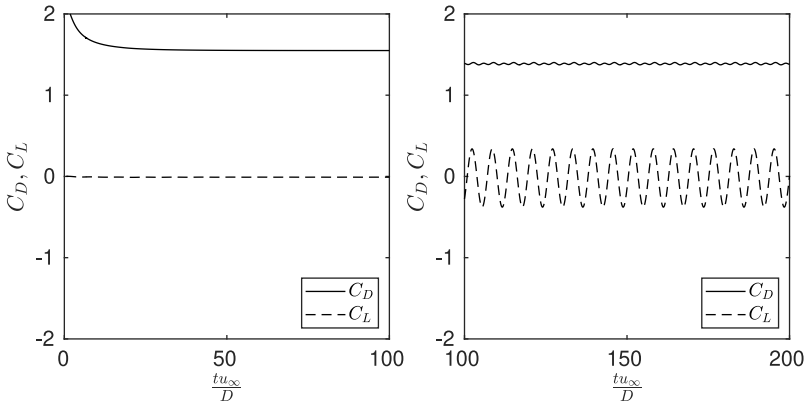



**Fig. 9.** Temporal evolution of the drag $C_D$ and lift $C_L$ coefficients for the flow over a stationary cylinder at different $Re$.

**Table 3**
Comparison of the vortices center location for the lid-driven cavity with embedded cylinder case for $Re = 1000$. The vortices center $(x_1, y_1)$, $(x_2, y_2)$, and $(x_3, y_3)$ correspond to the bottom left corner, bottom right corner, and top right of the cylinder respectively.

| | $(x_1, y_1)$ | $(x_2, y_2)$ | $(x_3, y_3)$ |
|---|---|---|---|
| Cai et al. (2017) [15] | (0.0789, 0.0720) | (0.8852, 0.1063) | (0.6942, 0.6881) |
| Present | (0.0787, 0.0717) | (0.8847, 0.1062) | (0.6921, 0.6873) |

The contours of the velocity magnitude and the vorticity are shown in Figs. 5(a) and 5(b) respectively. The results are very similar to the previous works of [15,23]. The streamlines are illustrated in Fig. 5(c). It is evident that three vortices emerged in the flow, located at the bottom left and right corners, and the third one located on the top right of the embedded cylinder. The location of the center of each vortex is compared in Table 3 with the data reported in [15]. A very good agreement can be found.

A quantitative comparison for the velocity profiles is conducted with previous IBM data presented in [15,23], for the velocity components $(u, v)$ in the vertical $(x = 0.5)$ and horizontal $(y = 0.5)$ centerlines of the cavity. The data obtained is shown in Fig. 6. Excellent agreement with previous studies is again obtained.

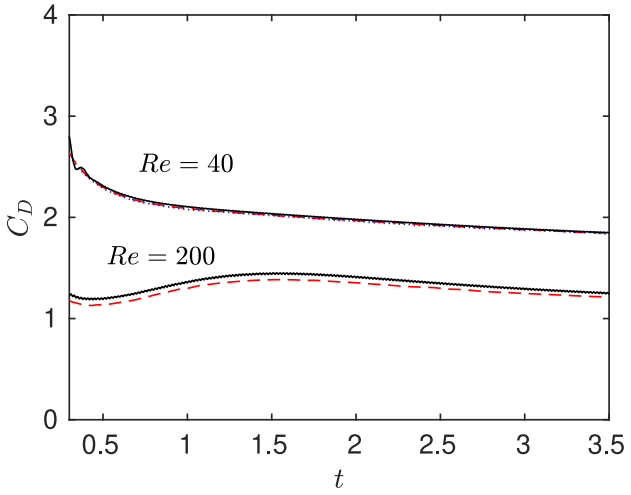
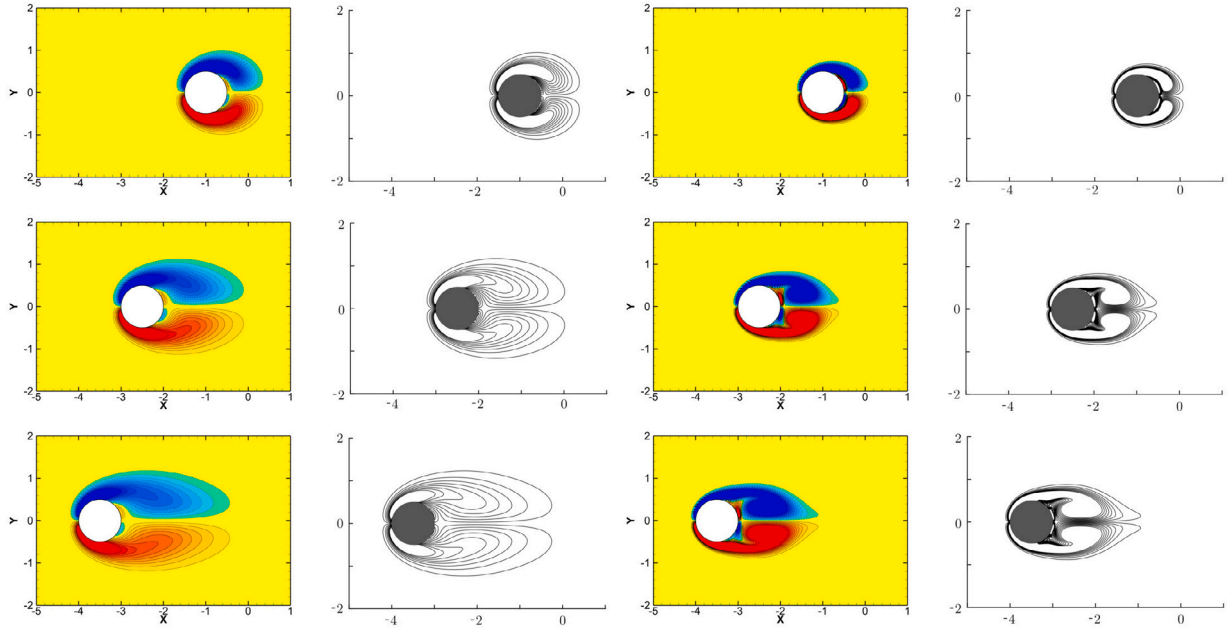



**Fig. 10.** Temporal evolution of the drag $C_D$ coefficient for the flow over an impulsively started cylinder at $Re = 40$ and $Re = 200$: (——) present results, (· · · · ·) data reported in [15], (– – –) data reported in [13].

### 5.3. Flow over a stationary cylinder

The benchmark problem of the steady and unsteady flows over a stationary circular cylinder is also considered. For this test case, Reynolds numbers $Re = 40$ and 100 are simulated, where $Re = u_\infty D/\nu$, having $u_\infty$ as the freestream inlet velocity and $D$ the cylinder diameter. The cylinder diameter and inlet velocity are fixed to 1, while the kinematic viscosity is varied to obtain the simulated $Re$ numbers. A large Eulerian squared domain $\Omega_f = [-20, 40] \times [-30, 30] = 60D \times 60D$ is used to minimize the lateral boundaries confinement effect, with an Eulerian grid resolution $h = 0.03$. The cylinder is located at the center of the Eulerian domain where the numbers of uniformly distributed Lagrangian markers forming $\Gamma_s$ is set to 104 with $\Delta s/h \approx 1$. A time-step of 0.006 is applied. The 3-point-width kernel function $\phi_1$ is selected for the data mapping between the two grids.

A Dirichlet boundary condition $(u_\infty = 1, v_\infty = 0)$ is set on the inflow boundary, and a non-reflective advective boundary condition $(\partial \mathbf{u}/\partial t + u_n \partial \mathbf{u}/\partial \mathbf{n} = 0)$ is applied at the outflow boundary with $u_n$ being



(a) $Re = 40$ (b) $Re = 200$

**Fig. 11.** Instantaneous vorticity contours at three different time instances $t = 1$ (top), 2.5 (middle), and 3.5 (bottom) for the flow over an impulsively started cylinder at $Re = 40$ (a) and $Re = 200$ (b). The present results correspond to the colored contours and the results acquired by [13] correspond to the black/white contours. The present vorticity contours are varied from −3 (blue) to 3 (red).

**Table 4**
Comparison of the wake dimensions and the mean drag coefficient for the flow over a stationary cylinder at $Re = 40$.

| References | $l/D$ | $a/D$ | $b/D$ | $\theta(^\circ)$ | $\overline{C_D}$ |
|---|---|---|---|---|---|
| Tritton [30][a] | – | – | – | – | 1.585 |
| Coutanceau & Bouard [31][a] | 2.13 | 0.76 | 0.59 | 53.5 | – |
| Taira & Colonius [13] | 2.33 | 0.75 | 0.60 | 54.1 | 1.550 |
| Wang & Zhang [27] | 2.36 | 0.72 | 0.60 | 53.8 | 1.540 |
| Cai et al. [15] | 2.36 | 0.75 | 0.62 | 52.1 | 1.560 |
| Present | 2.24 | 0.78 | 0.61 | 52.0 | 1.550 |

[a] Experimental results.

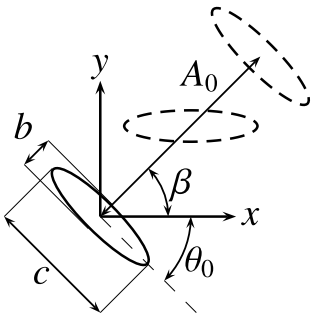



**Fig. 12.** Configuration of the flapping ellipse.

the advection velocity, i.e., the component of the velocity normal to the boundary. Slip boundary conditions are set to the lateral sides of the domain.

The instantaneous non-dimensional force coefficients and Strouhal number are defined as:

$$\text{Drag and lift coefficients: } C_D = \frac{F_D}{0.5\rho u_\infty{}^2 D} \text{ and } C_L = \frac{F_L}{0.5\rho u_\infty{}^2 D},$$

$$\text{Strouhal number: } \text{Str} = \frac{f_s D}{u_\infty},$$

where the fluid density $\rho$ is set to 1 since we are working with incompressible flows, $f_s$ is the natural vortex shedding frequency, and $(F_D, F_L)$ are the respective drag and lift forces exerted on the cylinder's immersed boundary by the fluid. They are computed by summing the singular Lagrangian forces located on each Lagrangian marker:

$$\begin{pmatrix} F_D \\ F_L \end{pmatrix} = -\rho \sum_{n=1}^{N_{Lag}} \mathbf{F}_{IB}(\mathbf{X}_n) W_n. \tag{35}$$

For $Re = 40$, the flow pattern is expected to be symmetric and steady, where a recirculation region is formed in the wake of the cylinder having the following dimensions: $l$, the wake streamwise length, $a$, the horizontal distance from the cylinder boundary to the wake vortex center, $b$ the vertical distance between the center of the two wake vortices, and $\theta$ the separation angle measured from the $x$-axis. The obtained pressure contours, vorticity contours, and streamlines for the steady flow at one time-instant are displayed in Fig. 7.

It can be noted that the flow profile closely matches those documented in the literature. In addition, the symmetric wake dimensions are compared with previous experimental and numerical results, showing good consistency, see Table 4. The drag and lift coefficients are plotted in Fig. 9(a), and the steady-state drag value is also compared with values extracted from the literature in Table 4.

At $Re = 100$, the flow pattern becomes unsteady and exhibits the famous von Kármán vortex street, also known as periodic vortex shedding. The pressure and vorticity contours are shown in Fig. 8. The vortex shedding leads to time-varying fluctuations of the drag and lift coefficients, they can clearly be seen in Fig. 9(b). Different force quantities obtained for the current simulations, including the mean drag coefficient, the drag and lift fluctuations, and the Strouhal number, are compared against experimental and numerical results from the literature at this $Re$ number. These quantities are computed after a periodic state has been achieved by the vortex shedding. A close agreement can again be established when comparing with previous results as shown in Table 5.

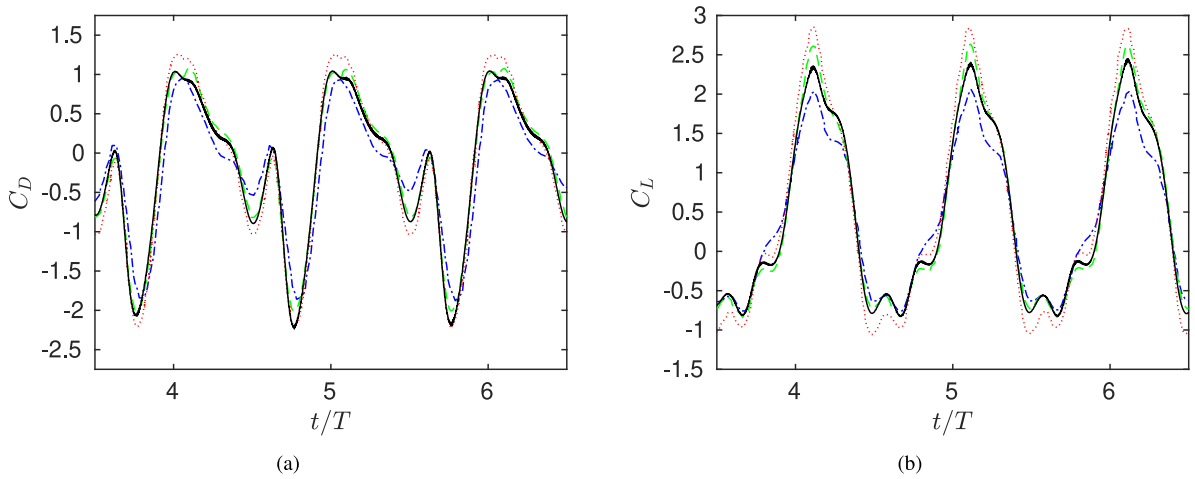



**Fig. 13.** Temporal evolution of the drag $C_D$ (a) and lift $C_L$ (b) coefficients for the flow over a flapping ellipse at $Re = 157$: (——) present results, (· · · · · ·) immersed boundary method of [15], (- - -) immersed boundary method of [34], (-·-·-) body-conformal mesh method [35].

**Table 5**
Comparison of the mean drag coefficient, the drag and lift fluctuations amplitude, and the Strouhal number for the flow over a stationary cylinder at $Re = 100$.

| References | $\overline{C_D}$ | $C'_D$ | $C'_L$ | $Str$ |
|---|---|---|---|---|
| Williamson [32][a] | – | – | – | 0.166 |
| Braza et al. [33] | 1.364 | ±0.015 | ±0.250 | 0.160 |
| Uhlmann [8] | 1.453 | ±0.011 | ±0.339 | 0.169 |
| Ji et al. [14] | 1.402 | ±0.010 | ±0.349 | 0.167 |
| Cai et al. [15] | 1.366 | ±0.010 | ±0.342 | 0.160 |
| Cheylan et al. [10] | 1.408 | – | ±0.330 | 0.166 |
| Present | 1.387 | ±0.011 | ±0.359 | 0.160 |

[a] Experimental results.

### 5.4. Flow over an impulsively started cylinder

The flow over an accelerated cylinder in a stagnant fluid is considered at two different Reynolds numbers $Re = 40$ and 200. The circular cylinder of diameter $D = 1$ is located initially at $t = 0$ at the center of an Eulerian domain $\Omega_f = [-16.5, 13.5] \times [-15, 15] = 30D \times 30D$ with a resolution $h = 0.02$, to compare with results summarized in [13]. At $t > 0$ the cylinder is impulsively set into a translational motion with a constant speed $U_0 = -1$ and having $Re$ defined by $U_0 D/\nu$. A no-slip boundary condition is applied to all Eulerian domain boundaries. The time-step is fixed to 0.004 for all simulations and the 5-point-width kernel function is used to reduce the non-physical spurious force oscillations. The simulation time is advanced until a value of $t = 3.5$ is reached. The circular cylinder boundary $\Gamma_s$ is uniformly covered by 156 Lagrangian markers.

The temporal evolution of the drag coefficient at $Re = 40$ and 200 are plotted in Fig. 10 and compared to previous numerical data obtained with the immersed boundary methods developed in [13,15]. A good agreement is obtained for $Re = 40$ when compared to the literature. In contrast, for $Re = 200$, the results of the drag coefficient are found to be almost 5% larger than in [13] at the beginning of the simulation. This error decreases to a value less than 1% at the end of the simulation time.

Snapshots of the vorticity field at different time instances $t = 1$, 2.5, and 3.5 are compared to the ones obtained by [13] for $Re = 40$ and 200 in Figs. 11(a) and 11(b), respectively. The vorticity contours are in excellent agreement with those obtained in [13].

### 5.5. Flow over a flapping ellipse

The flow induced by a non-circular object in a quiescent fluid is simulated, where the immersed boundary is described by a 2D ellipse. The flapping of the ellipse is subject to both translational and rotational motions governed by the following equations:

$$A(t) = \frac{A_0}{2}\left[cos\left(\frac{2\pi t}{T}\right) + 1\right], \tag{36}$$

$$\theta(t) = \frac{\theta_0}{2}\left[1 - sin\left(\frac{2\pi t}{T} + \phi_0\right)\right], \tag{37}$$

where $A$ is the time-dependent flapping distance due to motion, $\theta$ is the time-dependent angle of attack of the ellipse, $A_0$ and $\theta_0$ designate the translational and rotational amplitudes, respectively, $\beta$ is the stroke angle, $\phi_0$ is the phase difference between the translational and rotational movements and $T$ is the flapping period. The configuration of the moving ellipse is described in Fig. 12, having $c$ as the major axis and $b$ as the minor axis of the ellipse. The flow is characterized by the maximum velocity of translational motion along the flapping path computed as $U_{max} = \pi A_0/T$. The flapping period is defined by $T = \pi A_0/c$, and the Reynolds number $Re = U_{max} c/\nu$. The simulation is performed at $Re = 157$, $c = 1$, $c/b = 4$, $A_0 = 2.5$, $\theta_0 = \pi/2$, $\beta = \pi/3$, $\phi_0 = 0$, $U_{max} = 1$, and $T = 2.5\pi$, to compare with previously published numerical studies from the literature [15,34,35].

The Eulerian domain is defined by $\Omega_f = [-10, 10] \times [10, 10] = 20D \times 20D$, having the grid resolution $h = 0.02$. Neumann boundary conditions are set for both pressure ($\partial p/\partial \mathbf{n} = 0$) and velocity ($\partial \mathbf{u}/\partial \mathbf{n} = 0$) for all domain boundaries. A constant time-step of $\Delta t = 0.004$ is applied for this simulation, and $\phi_2$ is used as the kernel function to minimize the non-physical spurious force oscillations. The ellipse is covered by 106 Lagrangian markers uniformly distributed over $\Gamma_s$.

A quantitative validation for the forces generated by the flapping ellipse is carried out, by comparing the time history of the instantaneous drag and lift coefficients in Fig. 13 with previous studies computed using a body-conformal mesh method [35] and body non-conforming immersed boundary methods [15,34]. The data obtained with the present DF-IBM agree well with the literature, even when performed on a smaller Eulerian domain. It is worth noting that the interior of the immersed boundary is treated as part of the fluid Eulerian domain. Ergo, an internal and external fluid are situated around the immersed boundary. The drag and lift forces exerted on the hovering ellipse are influenced by this internal fluid (or mass). Once the internal fluid is moved, the inertia created by this accelerating mass should be added to the force equations to compensate for the accelerating

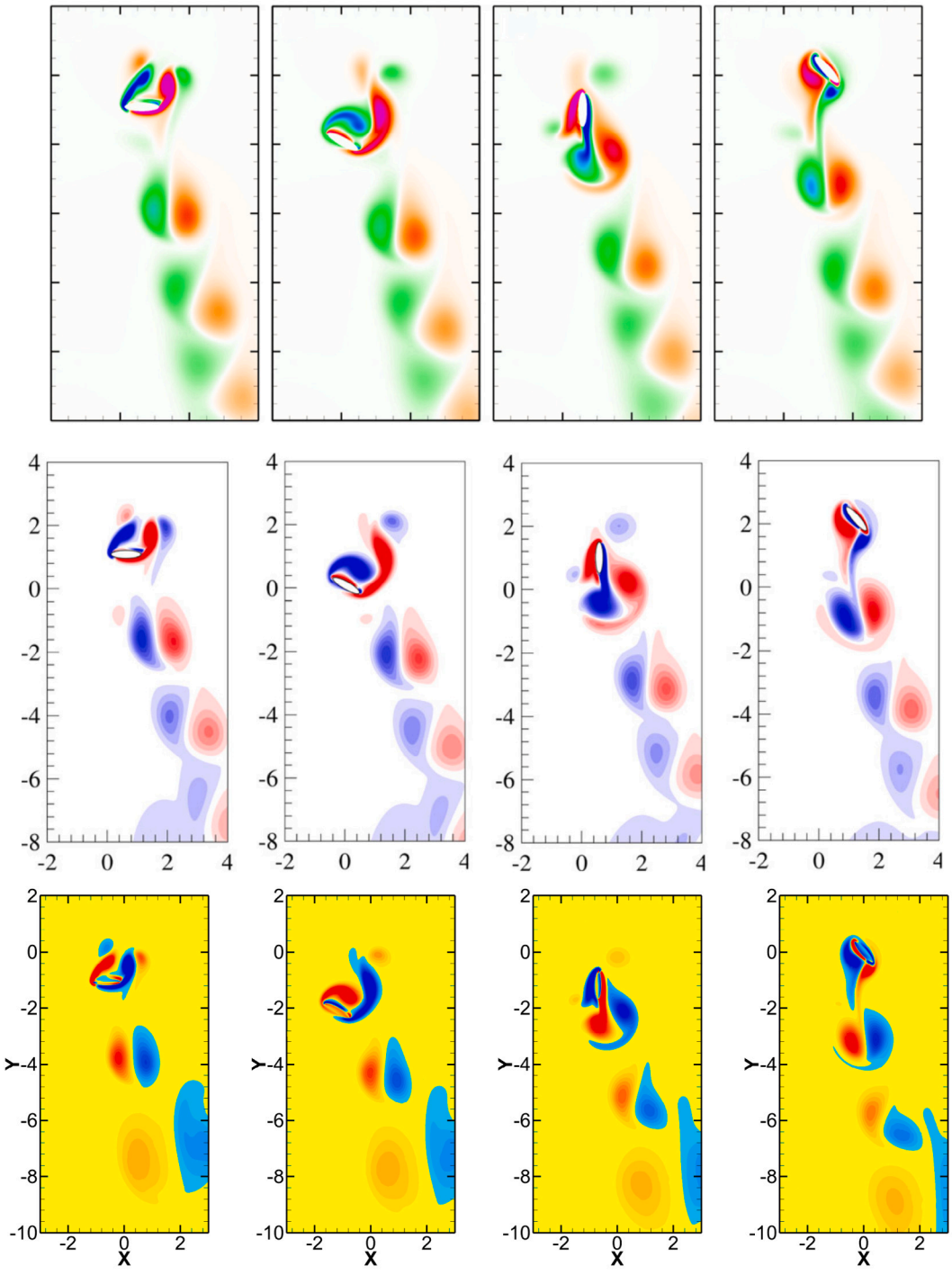



**Fig. 14.** Instantaneous vorticity contours at four different period instances $t/T$ for the flow over a flapping ellipse at $Re = 157$. The present results are placed at the bottom, the results corresponding to [15] are placed in the middle, and the results corresponding to [34] are placed at the top. The present vorticity contours are varied from −3 (blue) to 3 (red).

virtual fluid inside the immersed boundary [36]. Therefore, unlike the computation of drag and lift forces for flow over a stationary cylinder shown by Eq. (35), the forces for moving solid immersed boundaries are computed using the following approach:

$$\begin{pmatrix} F_D \\ F_L \end{pmatrix} = -\rho \sum_{n=1}^{N_{Lag}} \mathbf{F}_{IB}(\mathbf{X}_n, t) W_n + \rho V \frac{d\mathbf{U}_{IB}(\mathbf{X}_n, t)}{dt}, \tag{38}$$

where $V$ is the internal volume occupied by the immersed boundary.

Fig. 14 compares the instantaneous vorticity field produced by the flapping ellipse at four different instances of a single flapping period, i.e. $t/T = 0.25$, 0.44, 0.74, and 0.99. These contours are similar to those obtained by [15,34].

## 6. Conclusions

A new fast implicit direct-forcing immersed boundary method (FIDF-IBM) is presented for the numerical simulations of incompressible flows

around arbitrarily moving rigid structures. The method employs the operator splitting strategy from the original PISO algorithm to maintain a divergence-free condition on the Eulerian domain and enforce the ns/np boundary condition on the Lagrangian domain.

A modified pressure Poisson equation (PPE) is derived, incorporating the boundary force where the ns/np condition is already satisfied, yielding a coefficient matrix similar to a standard PPE. This enables the direct use of existing fast iterative PPE solvers.

This work shows a significant reduction in the computational cost of satisfying the ns/np boundary condition by combining Lagrangian weight methods that improve the reciprocity of interpolation-spreading operators, with the implicit formulation of the DF-IBM. Moreover, by using a suitable Lagrangian boundary force inheritance in the DF-IBM algorithm further enhances the performance of the proposed scheme.

The current algorithm can effectively handle stationary and moving immersed boundaries in both translational and rotational motions, for circular and non-circular solid structures. The computed results align well with published experimental and numerical data.

## CRediT authorship contribution statement

**Elias Farah:** Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization. **Abdellatif Ouahsine:** Validation, Supervision, Project administration, Methodology, Investigation, Conceptualization. **Patrick Verdin:** Writing – review & editing, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization.

## Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Elias Farah reports financial support was provided by UK Engineering and Physical Sciences Research Council (EPSRC). If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This research was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) (Grant No. EP/T518104/1, Project Reference No. 2676291). The computations were performed on the platform PILCAM2 from the Université de Technologie de Compiègne, Alliance Sorbonne Université. The data that support the findings of this study are openly available in CERES Research at https://doi.org/10.57996/cran.ceres-2760.

## Appendix. $L_2$ error norm

The $L_2$ norm of the error for any variable $x$ is defined as:

$$\|E_x\|_2 = \sqrt{\frac{1}{N_x N_y}\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\left|x_{i,j} - \hat{x}_{i,j}\right|^2}, \tag{A.1}$$

where $x_{i,j}$ is the value obtained numerically, and $\hat{x}_{i,j}$ is the analytical or refined solution in case no exact solution exists. $N_x$ and $N_y$ are the total number of Eulerian grid points in the $x$ and $y$ directions, respectively.

## Data availability

The data that support the findings of this study are openly available in CERES Research at https://doi.org/10.57996/cran.ceres-2760.

## References

[1] Peskin CS. Flow patterns around heart valves: A numerical method. J Comput Phys 1972;10(2):252–71. http://dx.doi.org/10.1016/0021-9991(72)90065-4.
[2] Sotiropoulos F, Yang X. Immersed boundary methods for simulating fluid-structure interaction. Prog Aerosp Sci 2014;65:1–21. http://dx.doi.org/10.1016/j.paerosci.2013.09.003.
[3] Huang W-X, Tian F-B. Recent trends and progress in the immersed boundary method. Proc Inst Mech Eng Part C J Mech Eng Sci 2019;233(23–24):7617–36. http://dx.doi.org/10.1177/0954406219842606.
[4] Kim W, Choi H. Immersed boundary methods for fluid-structure interaction: A review. Int J Heat Fluid Flow 2019;75:301–9. http://dx.doi.org/10.1016/j.ijheatfluidflow.2019.01.010.
[5] Griffith BE, Patankar NA. Immersed methods for fluid-structure interaction. Annu Rev Fluid Mech 2020;52:421–48. http://dx.doi.org/10.1146/annurev-fluid-010719-060228.
[6] Verzicco R. Immersed boundary methods: Historical perspective and future outlook. Annu Rev Fluid Mech 2023;55:129–55. http://dx.doi.org/10.1146/annurev-fluid-120720-022129.
[7] Mittal R, Seo JH. Origin and evolution of immersed boundary methods in computational fluid dynamics. Phys Rev Fluids 2023;8(10):100501. http://dx.doi.org/10.1103/PhysRevFluids.8.100501.
[8] Uhlmann M. An immersed boundary method with direct forcing for the simulation of particulate flows. J Comput Phys 2005;209(2):448–76. http://dx.doi.org/10.1016/j.jcp.2005.03.017.
[9] Luo K, Wang Z, Fan J, Cen K. Full-scale solutions to particle-laden flows: Multi-direct forcing and immersed boundary method. Phys Rev E 2007;76(6):066709. http://dx.doi.org/10.1103/PhysRevE.76.066709.
[10] Cheylan I, Fringand T, Jacob J, Favier J. Analysis of the immersed boundary method for turbulent fluid-structure interaction with lattice Boltzmann method. J Comput Phys 2023;492:112418. http://dx.doi.org/10.1016/j.jcp.2023.112418.
[11] Kempe T, Fröhlich J. An improved immersed boundary method with direct forcing for the simulation of particle laden flows. J Comput Phys 2012;231(9):3663–84. http://dx.doi.org/10.1016/j.jcp.2012.01.021.
[12] Breugem W-P. A second-order accurate immersed boundary method for fully resolved simulations of particle-laden flows. J Comput Phys 2012;231(13):4469–98. http://dx.doi.org/10.1016/j.jcp.2012.02.026.
[13] Taira K, Colonius T. The immersed boundary method: A projection approach. J Comput Phys 2007;225(2):2118–37. http://dx.doi.org/10.1016/j.jcp.2007.03.005.
[14] Ji C, Munjiza A, Williams JJR. A novel iterative direct-forcing immersed boundary method and its finite volume applications. J Comput Phys 2012;231(4):1797–821. http://dx.doi.org/10.1016/j.jcp.2011.11.010.
[15] Cai S-G, Ouahsine A, Favier J, Hoarau Y. Moving immersed boundary method. Internat J Numer Methods Fluids 2017;85(5):288–323. http://dx.doi.org/10.1002/fld.4382.
[16] Gsell S, D'Ortona U, Favier J. Explicit and viscosity-independent immersed-boundary scheme for the lattice Boltzmann method. Phys Rev E 2019;100(3):033306. http://dx.doi.org/10.1103/PhysRevE.100.033306.
[17] Gsell S, Favier J. Direct-forcing immersed-boundary method: A simple correction preventing boundary slip error. J Comput Phys 2021;435:110265. http://dx.doi.org/10.1016/j.jcp.2021.110265.
[18] Pinelli A, Naqavi IZ, Piomelli U, Favier J. Immersed-boundary methods for general finite-difference and finite-volume Navier–Stokes solvers. J Comput Phys 2010;229(24):9073–91. http://dx.doi.org/10.1016/j.jcp.2010.08.021.
[19] Roma AM, Peskin CS, Berger MJ. An adaptive version of the immersed boundary method. J Comput Phys 1999;153(2):509–34. http://dx.doi.org/10.1006/jcph.1999.6293.
[20] Bao Y, Kaye J, Peskin CS. A Gaussian-like immersed-boundary kernel with three continuous derivatives and improved translational invariance. J Comput Phys 2016;316:139–44. http://dx.doi.org/10.1016/j.jcp.2016.04.024.
[21] Yang X, Zhang X, Li Z, He G-W. A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulationsg. J Comput Phys 2009;228(20):7821–36. http://dx.doi.org/10.1016/j.jcp.2009.07.023.
[22] Griffith BE, Luo X. Hybrid finite difference/finite element immersed boundary method. Int J Numer Methods Biomed Eng 2017;33(12):e2888. http://dx.doi.org/10.1002/cnm.2888.
[23] Goncharuk K, Oshri O, Feldman Y. The immersed boundary method: A SIMPLE approach. J Comput Phys 2023;487:112148. http://dx.doi.org/10.1016/j.jcp.2023.112148.
[24] Constant E, Favier J, Meldi M, Meliga P, Serre E. An immersed boundary method in openfoam: Verification and validation. Comput & Fluids 2017;157:55–72. http://dx.doi.org/10.1016/j.compfluid.2017.08.001.
[25] Askarishahi M. Immersed-boundary/soft-sphere method for particle–particle-fluid interaction in a viscous flow: An openfoam solver. Adv Powder Technol 2023;34(11):104204. http://dx.doi.org/10.1016/j.apt.2023.104204.
[26] Farah E, Ouahsine A, Verdin PG. An improved implicit direct-forcing immersed boundary method (DF-IBM) around arbitrarily moving rigid structures. Phys Fluids 2024;36(10):103618. http://dx.doi.org/10.1063/5.0231218.

[27] Wang S, Zhang X. An immersed boundary method based on discrete stream function formulation for two- and three-dimensional incompressible flows. J Comput Phys 2011;230(9):3479–99. http://dx.doi.org/10.1016/j.jcp.2011.01.045.

[28] Jasak H, Weller HG, Gosman AD. High resolution NVD differencing scheme for arbitrarily unstructured meshes. Internat J Numer Methods Fluids 1999;31(2):431–49. http://dx.doi.org/10.1002/(SICI)1097-0363(19990930)31:2<431::AID-FLD884>3.0.CO;2-T.

[29] Issa RI. Solution of the implicitly discretised fluid flow equations by operator-splitting. J Comput Phys 1985;62(1):40–65. http://dx.doi.org/10.1016/0021-9991(86)90099-9.

[30] Tritton DJ. Experiments on the flow past a circular cylinder at low Reynolds numbers. J Fluid Mech 1959;6(4):547–67. http://dx.doi.org/10.1017/S0022112059000829.

[31] Coutanceau M, Bouard R. Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation. Part 1. Steady flow. J Fluid Mech 1977;79(2):231–56. http://dx.doi.org/10.1017/S0022112077000135.

[32] Williamson CHK. Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low Reynolds numbers. J Fluid Mech 1989;206:579–627. http://dx.doi.org/10.1017/S0022112089002429.

[33] Braza M, Chassaing P, Minh H Ha. Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder. J Fluid Mech 1986;165:79–130. http://dx.doi.org/10.1017/S0022112086003014.

[34] Yang J, Stern F. A simple and efficient direct forcing immersed boundary framework for fluid–structure interactions. J Comput Phys 2012;231(15):5029–61. http://dx.doi.org/10.1016/j.jcp.2012.04.012.

[35] Wang ZJ. Two dimensional mechanism for insect hovering. Phys Rev Lett 2000;85(10):2216–9. http://dx.doi.org/10.1103/PhysRevLett.85.2216.

[36] Suzuki K, Inamuro T. Effect of internal mass in the simulation of a moving body by the immersed boundary method. Comput & Fluids 2011;49(1):173–87. http://dx.doi.org/10.1016/j.compfluid.2011.05.011.



https://dspace.org/

# Fast implicit direct-forcing immersed boundary method (FIDF-IBM)

**Farah, Elias; Ouahsine, Abdellatif; Verdin, Patrick G.**

**2025-10**
Elsevier
https://doi.org/10.1016/j.enganabound.2025.106345
Farah E, Ouahsine A, Verdin P. (2025) Fast implicit direct-forcing immersed boundary method (FIDF-IBM). Engineering Analysis with Boundary Elements, Volume 179, Part B, October 2025, Article number 106345

Downloaded from a DSpace repository. https://dspace.org/