

Article

POF24-AR-FSI2024-08256

An improved implicit direct-forcing immersed boundary method (DF-IBM) around arbitrarily moving rigid structures

E. Farah,^{1,2} A. Ouahsine,¹ and P. G. Verdin²

¹*Université de Technologie de Compiègne, Alliance Sorbonne Université. CNRS, UMR 7337 Roberval, Centre de Recherche Royallieu, CS 60319, 60203 Compiègne, France.*

²*Cranfield University, Energy and Sustainability Department. College Rd, Cranfield, Wharley End, Bedford MK43 0AL, United Kingdom.*

(*Electronic mail: elias.farah@utc.fr, ouahsine@utc.fr)

(Dated: 26 September 2024)

An improved implicit direct-forcing immersed boundary method (DF-IBM) is presented for simulating incompressible flows around complex rigid structures undergoing arbitrary motion. The current approach harnesses the pressure implicit with splitting of operators (PISO) algorithm to handle the fluid-solid system's dual constraints in a segregated manner. As a result, the divergence-free condition is preserved throughout the Eulerian domain and the no-slip velocity boundary condition is exactly enforced on the immersed boundary. A new pressure Poisson equation (PPE) is derived, incorporating the boundary force where the no-slip condition is already met, enabling the use of fast iterative PPE solvers without modifications. The improvement involves integrating Lagrangian weight methods having better reciprocity over the IBM-related linear operators with the implicit formulation. An additional force initialization scheme is introduced to further boost the algorithm's performance. The method's accuracy, efficiency, and capability are verified through various stationary and moving immersed boundary cases. The results are validated against experimental and numerical data from the literature. The proposed improvements seamlessly integrate into existing incompressible fluid solvers with minimal adjustments to the original system equations, highlighting their ease of implementation.

Article

I. INTRODUCTION

Complex fluid-structure interaction (FSI) problems are usually simulated using conventional body-conforming mesh techniques. These methods are considered computationally expensive once body motion and deformation are present for complex geometries. This necessitates the development of sophisticated techniques and strategies to avoid critical mesh distortion and strong deformation, while preserving discretization accuracy^{1–3}.

Body non-conforming meshes are contemplated as an alternative, that facilitates the usage of simple Cartesian meshes. This approach notably simplifies the heavy mesh generation procedure, by eliminating mesh deformation and distortions, and the re-meshing procedures in case of moving structures. Paving the way for a new era of non-conforming meshing techniques, the most widely used example in recent years is the immersed boundary method (IBM)⁴.

Developed by Peskin in 1972⁵, the IBM fundamental idea is to use two separate domains. The first is fixed and non-deformable, labeled as the Eulerian domain, whose nodes do not conform the solid geometry, and where the fluid Navier-Stokes equations are solved. The second, labeled as the Lagrangian domain, represents the solid immersed boundary using a discrete set of points, named Lagrangian markers, which are used to track the information at the embedded structure border. These two domains eliminate the meshing-related issues when using the Arbitrary Lagrangian-Eulerian (ALE) method in the body-conforming realm^{6,7}. As a result, separate interaction equations were created to transfer the data between both domains, named interpolation and spreading operators. The novelty of IBM lies in the incorporation of a fictitious body force, named boundary force hereupon, in the governing equations as a source term to compensate for the real effects of the solid immersed boundary.

In the present work, the direct-forcing IBM (DF-IBM) framework of Uhlmann⁸ is adopted, due to its widespread among researchers, stability, versatility, and ease of implementation in any fluid solver, owing to its minimal requirement for integration in the discretized form of the Navier-Stokes equations. The originality of the DF-IBM is that it eliminates the constitutive relation restrictions for rigid bodies found in the classical IBM⁵, which was primarily designed for elastically deformable boundaries.

The approach of Uhlmann⁸, termed as the explicit DF-IBM, failed remarkably to impose the no-slip boundary condition on the immersed boundary as reported in Refs. 9–11. Therefore a multi-direct forcing approach was introduced, dubbed here multi-DF-IBM. The idea is to perform

Article

multiple direct-forcing loops to improve the no-slip boundary condition. It was agreed in Refs. 10 and 11 that three loops are enough to reduce the error on the immersed boundary while maintaining a good computational efficiency of the algorithm. This approach is also regarded as explicit, since a prescribed number of direct-forcing loops is implemented in the algorithm. If the exact imposition of the no-slip boundary constraint is of interest, implicit IBMs are constructed for such a purpose. Iterations of the whole IBM system are required until reaching a certain tolerance specified by the user, which approximates the boundary force computations by using a closer value to the final velocity field. A fully implicit IBM was developed in Ref. 12 for the projection method, referred to as the immersed boundary projection method (IBPM). The main idea behind this method is to couple the velocity field to both the pressure field and boundary force to form a set of Lagrange multipliers to enforce the divergence-free and no-slip boundary constraints simultaneously. A new form for the pressure Poisson equation (PPE) is obtained with the pressure and boundary force considered as unknowns, solved simultaneously in a single projection step. However, it was revealed by Ji et al.¹³, that the IBPM suffers from convergence problems, when a Lagrangian marker is situated close enough to an Eulerian grid point, contributing to a larger condition number of the coefficient matrix of the new PPE, due to the singular property of the interpolation and spreading operators. Hence to avoid such convergence problems, Ji et al.¹³ proposed to iterate each constraint alone, so that the coefficient matrix is never modified, having the no-slip boundary constraint iterated inside the continuity constraint, which is taken as the outer loop, resulting in a drastic increase in the computational effort. Although expensive, this approach is nevertheless interesting as it provides simplicity and ease of implementation in any existing fluid solver code. A moving implicit IBM¹⁴, referred to as MIBM, was later developed to maintain an acceptable computational efficiency compared to the explicit IBM variants and keep the accuracy of implicit IBM. The key idea of MIBM, is that an additional equation related to the boundary force is derived and labeled as the moving force equation. Analogously to the IBPM approach¹², the boundary force in the MIBM is also regarded as a Lagrange multiplier to enforce the no-slip boundary condition, where the moving force equation is derived directly from the exact no-slip boundary condition, producing a smaller coefficient matrix that is well-conditioned compared to the one obtained in the IBPM. This method showed good results and improved performance. However, its increased implementation complexity and increased computational memory requirements to store a new coefficient matrix for the moving force equation, especially when the immersed boundary geometry involves a much higher number of Lagrangian markers or multiple bodies, render it less

Article

appealing.

Another approach to improve the imposition of the no-slip boundary constraint, is through improving the mathematical reciprocity of the IBM-related interpolation and spreading operators which is translated directly to the computation of the Lagrangian weight used in the force spreading operator. It was proved in Refs. 15–17, that assigning the Lagrangian weight equal to the Eulerian weight, as originally suggested by Uhlmann⁸, would violate the reciprocal property of these operators, leading to a no-slip boundary error and fluid penetration inside the solid domain. To counter this problem, an appropriate method was proposed to determine the Lagrangian weight, see Pinelli et al.¹⁸, for instance. These authors presented a mathematical method to calculate the appropriate Lagrangian weight by solving an additional system of linear equations that improves the reciprocity of the interpolation and spreading equations. Another study¹⁶ introduced an analytical correction on the Lagrangian weight without the need to solve a linear system, which was later improved¹⁷ to account for the immersed boundary curvature.

In the current work, an improvement to the implicit DF-IBM variants is proposed, in the context of the finite volume approach. The pressure implicit with splitting of operators (PISO) algorithm is used to handle the pressure, velocity, and boundary force decoupling in the incompressible Navier-Stokes equations, and to enforce the no-slip boundary condition on the Lagrangian solid domain and the divergence-free condition on the Eulerian fluid domain in the solution process, following the splitting of operators strategy in the original PISO method. As the original PPE does not include the boundary force term, a modified PPE labeled mPPE is developed, having the boundary force already satisfying the no-slip boundary constraint, to avoid changing the coefficient matrix that might lead to convergence problems, and allowing the usage of fast iterative PPE solvers. In addition, to speed up the process to fulfill the no-slip boundary condition, the following points are applied i) different Lagrangian weight methods to improve the reciprocity of the IBM-related operators, and ii) the Lagrangian boundary force in the DF-IBM iterations is properly initialized.

In this work, the cell-centered incompressible Navier-Stokes fluid equations are discretized first, according to the finite volume approach in the open-source C++ toolbox OpenFOAM¹⁹ environment, version 7.0 of the OpenFOAM Foundation, having the $p - \mathbf{U}$ coupling handled by the PISO algorithm. The methodology behind DF-IBM is then presented, followed by a detailed explanation of the proposed DF-IBM algorithm and a comparison with previous algorithms. Afterward, numerical simulations are conducted to test the current implementation against published results from the literature. Finally, conclusions are drawn in the final section of this paper.

Article

II. NAVIER-STOKES EQUATIONS CELL-CENTERED FINITE VOLUME METHOD DISCRETIZATION

The fluid solver is described by the dimensional incompressible viscous fluid Navier-Stokes equations in the conservative form:

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

where the additional source term $\mathbf{f}$ [m/s^2] is the immersed boundary force also called boundary force term. The resulting dimension of every term is force per unit volume per density [m/s^2], with ν [m^2/s] the effective molecular kinematic viscosity and p [m^2/s^2] the kinematic pressure.

A. Finite volume method discretization

Each term of the Navier-Stokes equations is discretized in space using the Gauss theorem to reduce volume integrals to surface integrals, and then using the midpoint rule to transform the surface integrals into a discrete sum over all faces of the control volume. This yields the below algebraic equation, also called the discrete momentum equation:

$$\left(\frac{\partial \mathbf{u}_P}{\partial t} \right)^{n+1} + \sum_f \phi_f^n \mathbf{u}_f^{n+1} - \sum_f \nu_f (\nabla \mathbf{u})_f^{n+1} \cdot \mathbf{S}_f = -(\nabla p)_P^n V_P + \mathbf{f}_P^{n+1} V_P, \quad (2)$$

where $\mathbf{S}_f$ is the face area normal vector between the current control volumes P and N , and is equal to $\mathbf{S}_f = A_f \mathbf{n}_f$, having A_f the surface area of the face f and $\mathbf{n}_f$ the normal vector to the face f .

The face-center values of all explicit quantities $\phi_f^n = \mathbf{u}_f^n \cdot \mathbf{S}_f$ represent the face volumetric flux across the cell-face where the divergence-free condition is satisfied. They are computed from their neighboring cell-center values, as shown in Fig. 1 via a second-order linear interpolation, also called central differencing (CD) interpolation scheme. For Cartesian uniform meshes as in our case, i.e., when the mesh is formed by structured hexahedral, orthogonal, and equally spaced cells $\Delta x = \Delta y = h = |\mathbf{d}|$ as shown in Fig. 1, a special case arises for the linear interpolation where the approximation reduces to an arithmetic average between the owner (P) and the neighboring (N) control volumes.

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 scheme²⁰.

Article

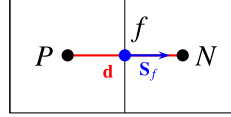


FIG. 1: Face area normal vector $\mathbf{S}_f$ in orthogonal meshes.

For the diffusion term, due to mesh uniformity and constant molecular viscosity, the face viscosity ν_f is the same as the laminar molecular kinematic viscosity ν . The face gradient term $(\nabla \mathbf{u})_f^{n+1} \cdot \mathbf{S}_f$ is treated using a central difference approximation of the first-order derivative.

For the pressure gradient term, once the linear algebraic equation is formed and ready to be solved, the cell-centered pressure gradient is computed explicitly in the predictor step of the PISO solution algorithm using the Gauss theorem, with the face values obtained from linear interpolation. Another pressure gradient term appears in the corrector step of the PISO solution algorithm where it is implicitly treated, and specifically in the PPE, as elaborated later in Section III of this paper. This term is similar to the diffusion term and is called the face normal pressure gradient.

The cell-centered boundary force is treated implicitly in the linear algebraic equation resulting from the discretized momentum equations, as explained in detail in Section III. Other face values of the boundary force term appear in the corrector step of the PISO algorithm inside the PPE. These face values are treated using a linear interpolation scheme.

For the solution's overall (space and time) accuracy to be second-order, all the individual terms in space and time have to be second-order. The temporal term is chosen to be discretized using a fully implicit three-level second-order scheme, also called BDF2 (Backward Differentiation Formula of order 2)²¹ in the literature.

Transforming face values to cell-centered values in Eq. (2) and dividing by V_P , we obtain the simplified linear algebraic equation:

$$a_P \mathbf{u}_P^{n+1} + \sum_N a_N \mathbf{u}_N^{n+1} = \mathbf{RHS}(\mathbf{u}_P^n, \mathbf{u}_P^{n-1}) - (\nabla p)_P^n + \mathbf{f}_P^{n+1}, \quad (3)$$

where a_P is the diagonal matrix coefficient of the owner cell P that contains all face-centered implicit contributions of the temporal, convection, and diffusion terms, a_N is the off-diagonal matrix coefficient of the neighboring cell N involving all face-centered implicit contributions of the convection and diffusion terms only, and $\mathbf{RHS}$ is the source term that includes all cell-centered explicit contribution of the transient term.

Article

B. $p - U$ coupling: PISO algorithm

In the unsteady incompressible Navier-Stokes equations, the pressure and velocity are dependent flow variables. They are coupled together through the continuity equation. As a result, solving the Navier-Stokes equations requires simultaneously determining both the pressure and velocity fields. The role of the pressure is to make sure the velocity is divergence-free.

There are two main families of methods for solving this coupled system of equations: fully coupled methods and segregated methods. The former is preferred in the computations of compressible flows while the latter has been widely employed in incompressible flow computations. In the segregated approach, the pressure and the velocity fields are decoupled numerically and then solved separately or consecutively.

Segregated implicit methods exist in OpenFOAM, such as the SIMPLE-type methods, including SIMPLE²², SIMPLEC²³, PISO²⁴, PIMPLE which combines the PISO and SIMPLE (SIMPLEC) algorithms. They are all based on the fundamental idea of using a predictor-corrector procedure.

The PISO algorithm is adopted in the present work for pressure-velocity decoupling. The built-in solver is modified to incorporate the solid immersed boundary using DF-IBM. The non-iterative nature of the PISO algorithm splits the operators into, one implicit momentum predictor step at the beginning of the time-step using a finite volume discretization of the Navier-Stokes by treating an estimated pressure field as known, followed by multiple explicit momentum corrector steps, usually 2 to 3²⁴. This approach enforces a divergence-free velocity field by the solution of the pressure equation using the projection method.

The implicit predictor step is represented by Eq. (3) without the inclusion of the boundary force term, and with the pressure solution from the previous time level. The equation for the predicted velocity field is re-written in the following form:

$$a_P \mathbf{u}_P^* + \sum_N a_N \mathbf{u}_N^* = \mathbf{RHS}(\mathbf{u}_P^n, \mathbf{u}_P^{n-1}) - (\nabla p)_P^n, \quad (4)$$

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

The multiple explicit corrector steps are derived here. Going back to Eq. (3), the neighboring velocities term is taken to the right-hand side of the equation, and the pressure gradient term is treated now implicitly as an unknown:

$$a_P \mathbf{u}_P^{n+1} = \mathbf{H}(\mathbf{u}_N^{n+1}, \mathbf{u}_P^n, \mathbf{u}_P^{n-1}) - (\nabla p)_P^{n+1} + \mathbf{f}_P^{n+1}, \quad (5)$$

Article

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

$$\mathbf{H}(\mathbf{u}_N^{n+1}, \mathbf{u}_P^n, \mathbf{u}_P^{n-1}) = -\sum_N a_N \mathbf{u}_N^{n+1} + \mathbf{RHS}(\mathbf{u}_P^n, \mathbf{u}_P^{n-1}). \quad (6)$$

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. From this formulation, a new cell-centered velocity can be defined:

$$\mathbf{u}_P^{n+1} = \frac{\mathbf{H}(\mathbf{u}_N^*, \mathbf{u}_P^n, \mathbf{u}_P^{n-1})}{a_P} - \frac{1}{a_P} (\nabla p)_P^{n+1} + \frac{1}{a_P} \mathbf{f}_P^{n+1}. \quad (7)$$

The cell-face velocity $\mathbf{u}_f^{n+1}$ needed in the discretized continuity equation can be expressed by linearly interpolating the cell-center values of $\mathbf{u}_P^{n+1}$ to face-center values:

$$\mathbf{u}_f^{n+1} = \left(\frac{\mathbf{H}(\mathbf{u}_N^*, \mathbf{u}_P^n, \mathbf{u}_P^{n-1})}{a_P} \right)_f - \left(\frac{1}{a_P} \right)_f (\nabla p)_f^{n+1} + \left(\frac{1}{a_P} \right)_f \mathbf{f}_f^{n+1}. \quad (8)$$

Taking the divergence of Eq. (7), and enforcing the divergence-free condition on the new time level results in the continuity equation also known as the divergence-free condition:

$$\nabla \cdot \mathbf{u}_P^{n+1} = \sum_f \mathbf{u}_f^{n+1} \cdot \mathbf{S}_f = \sum_f \phi_f^{n+1} = 0, \quad (9)$$

where $\phi_f^{n+1} = \mathbf{u}_f^{n+1} \cdot \mathbf{S}_f$ is the divergence-free volumetric flux through the cell-face.

Now to obtain the discrete pressure equation also known as the PPE, the previous equation for the velocity Eq. (7) is substituted in the continuity Eq. (9):

$$\nabla \cdot \left(\frac{1}{a_P} (\nabla p)_P^{n+1} \right) = \nabla \cdot \left(\frac{\mathbf{H}(\mathbf{u}_N^*, \mathbf{u}_P^n, \mathbf{u}_P^{n-1})}{a_P} \right) + \nabla \cdot \left(\frac{1}{a_P} \mathbf{f}_P^{n+1} \right), \quad (10a)$$

and for the face values format,

$$\sum_f \left(\left(\frac{1}{a_P} \right)_f (\nabla p)_f^{n+1} \right) \cdot \mathbf{S}_f = \sum_f \left(\frac{\mathbf{H}(\mathbf{u}_N^*, \mathbf{u}_P^n, \mathbf{u}_P^{n-1})}{a_P} \right)_f \cdot \mathbf{S}_f + \sum_f \left(\left(\frac{1}{a_P} \right)_f \mathbf{f}_f^{n+1} \right)_f \cdot \mathbf{S}_f. \quad (10b)$$

Solving the modified PPE (mPPE) expressed above by Eq. (10b), the solution of the pressure equation p^{n+1} is obtained. It contains both the physical pressure field, which originates from the flow field, as well as the correction part, which is responsible for enforcing the mass continuity and compensating the error in the initial pressure field.

It is important to note that the pressure equation, referred to as mPPE, is the main problem in the numerical simulation process. Solving the pressure equation is considered the most time-consuming aspect of the entire simulation due to its high condition number.

Article

The face volumetric flux of the face velocity is corrected using the new pressure solution and the boundary force, knowing that the mPPE is derived from the divergence-free condition, the new cell-face volumetric flux also satisfies the divergence-free condition:

$$\phi_f^{n+1} = \mathbf{u}_f^{n+1} \cdot \mathbf{S}_f = \left(\left(\frac{\mathbf{H}(\mathbf{u}_N^*, \mathbf{u}_P^n, \mathbf{u}_P^{n-1})}{a_P} \right)_f - \left(\frac{1}{a_P} \right)_f (\nabla p)_f^{n+1} + \left(\frac{1}{a_P} \right)_f \mathbf{f}_f^{n+1} \right) \cdot \mathbf{S}_f. \quad (11)$$

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

$$\mathbf{u}_P^{n+1} = \frac{\mathbf{H}(\mathbf{u}_N^*, \mathbf{u}_P^n, \mathbf{u}_P^{n-1})}{a_P} - \frac{1}{a_P} (\nabla p)_P^{n+1} + \frac{1}{a_P} \mathbf{f}_P^{n+1}. \quad (12)$$

C. Navier-Stokes system of linear algebraic equations solvers

In the present work, the mPPE leads to a symmetrical coefficient matrix where it was found that the Conjugate Gradient (CG) iterative solver in conjunction with the Geometric Agglomerated Algebraic Multigrid (GAMG) as a preconditioner and the Gauss-Seidel smoother, is superior to all remaining solvers and preconditioners combinations in terms of computational efficiency. Instead, the Diagonal-based Incomplete LU factorization (DILU) preconditioner with the Bi-Conjugate Gradient Stabilized (BiCGStab) solver showed to be more computationally efficient for the momentum equation that resulted in a non-symmetrical coefficient matrix, due to the convective term linearization.

III. IMMersed BOUNDARY METHOD (IBM)

A. Mathematical formulation of DF-IBM

The formulation derived involves a mixture of variables from the underlying Eulerian and Lagrangian domains. The general mathematical formulation of the DF-IBM is derived as follows:

$$\nabla \cdot \mathbf{u} = 0, \quad (13a)$$

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) - \nabla \cdot (\nu \nabla \mathbf{u}) = -\nabla p + \mathbf{f}, \quad (13b)$$

$$\mathbf{U}_{IB}(\mathbf{X}, t) = \mathcal{I}[\mathbf{u}(\mathbf{x}, t)], \quad (13c)$$

$$\mathbf{f}(\mathbf{x}, t) = \mathcal{I}[\mathbf{F}_{IB}(\mathbf{X}, t)], \quad (13d)$$

Article

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

B. Velocity interpolation and force spreading operators

The data transfer between the Eulerian and Lagrangian grids is conducted in a two-way procedure using a smoothed approximation of the Dirac delta function δ also known as a distribution function, as shown in Fig. 2. The former takes an Eulerian input and generates a Lagrangian output, while the latter does the contrary.

1. Velocity interpolation operator:

The Eulerian velocity field is interpolated back to each Lagrangian marker $\mathbf{X}_n$ using the following formula:

$$\begin{aligned} \mathbf{U}(\mathbf{X}_n, t) &= \mathcal{I}[\mathbf{u}(\mathbf{x}_{i,j}, t)] \\ &= \sum_{\mathbf{x}_{i,j} \in \mathcal{D}_{\mathbf{X}_n}} \mathbf{u}(\mathbf{x}_{i,j}, t) \delta_h(\mathbf{x}_{i,j} - \mathbf{X}_n) \Delta V_{i,j}, \end{aligned} \quad (14)$$

where for Cartesian uniform meshes, the Eulerian weight $\Delta V_{i,j} = h^2$ in $2D$, with h being the distance between two Eulerian grid points such as $h = \Delta x = \Delta y$ [m]. $(\mathbf{x}_{i,j} \in \mathcal{D}_{\mathbf{X}_n})$ represents the collection of all neighboring Eulerian grid points that fall inside the support domain $\mathcal{D}_{\mathbf{X}_n}$ of the Lagrangian marker $\mathbf{X}_n$.

2. Force Spreading Operator:

The computed Lagrangian force $\mathbf{F}_{IB}$ needs to be spread back to the neighboring Eulerian grid points in the support domain of the Lagrangian marker $\mathbf{X}_n$:

$$\begin{aligned} \mathbf{f}(\mathbf{x}_{i,j}, t) &= \mathcal{S}[\mathbf{F}_{IB}(\mathbf{X}_n, t)] \\ &= \sum_{\mathbf{x}_n \in \mathcal{D}_{\mathbf{x}_{i,j}}} \mathbf{F}_{IB}(\mathbf{x}_n, t) \delta_h(\mathbf{x}_{i,j} - \mathbf{x}_n) W_n, \end{aligned} \quad (15)$$

Article

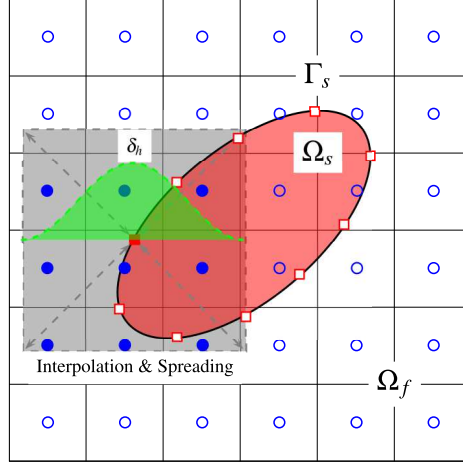


FIG. 2: Interpolation and spreading operators using a smoothed approximation of the Dirac delta function in the DF-IBM.

where W_n is the weight associated to the solid immersed boundary marker called the Lagrangian weight, 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 $\mathbf{x}_{i,j}$.

In addition, the immersed boundary is represented by a set of Lagrangian markers which are uniformly distributed over the boundary, with an arc length Δs [m], i.e., the distance between two consecutive Lagrangian markers, approximately equal to the Eulerian grid spacing h , in other terms $\Delta s/h \approx 1$.

C. Eulerian-Lagrangian mapping

In the present work, the conventional Dirac delta function was adopted in its discrete form. For Cartesian uniform meshes, the 2D approximation of the discrete Dirac delta function is given by:

$$\delta_h(\mathbf{x}_{i,j} - \mathbf{X}_n) = \delta_h(x_{i,j} - X_n) \delta_h(y_{i,j} - Y_n), \quad (16)$$

with

$$\delta_h(x_{i,j} - X_n) = \frac{1}{h} \phi(r_x) \quad \& \quad \delta_h(y_{i,j} - Y_n) = \frac{1}{h} \phi(r_y), \quad (17)$$

Article

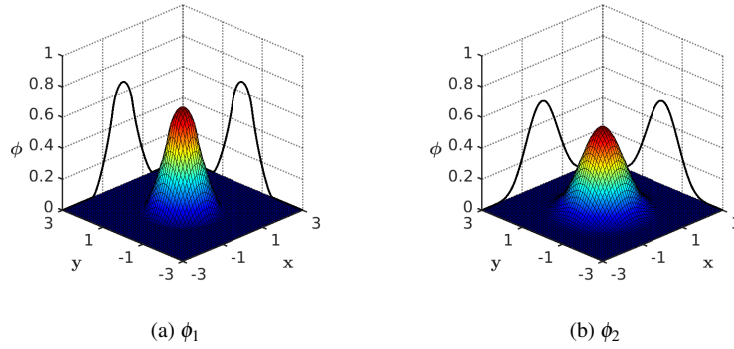


FIG. 3: One-dimensional and two-dimensional plots of the continuous kernel functions ϕ .

having

$$(r_x, r_y) = \left(\frac{x_{i,j} - X_n}{h}, \frac{y_{i,j} - Y_n}{h} \right), \quad (18)$$

where δ_h is expressed in $2D$ in $[1/m^2]$, ϕ is a non-dimensional continuous function named kernel function, and (r_x, r_y) denotes a non-dimensional distance between a Lagrangian marker and an Eulerian grid point. The kernel function scales to the mesh size h , i.e., the fluid grid spacing inside the Lagrangian support domain $\mathcal{D}_{\mathbf{X}_n}$ centered at each Lagrangian marker $\mathbf{X}_n = (X_n, Y_n)$. The Lagrangian support domain encloses all neighboring Eulerian grid points $\mathbf{x}_{i,j} = (x_{i,j}, y_{i,j})$ used for interpolating and spreading the velocity and boundary force, respectively. The continuous kernel function ϕ can take various forms. In the present numerical investigation, the 3-point-width function ϕ_1 developed in Ref. 25 and the 5-point-width function ϕ_2 developed in Ref. 26 are used, where the latter suppresses the non-physical force oscillations when a moving immersed boundary is simulated due to its high moment condition²⁷.

The former kernel functions are displayed in one-dimensional and two-dimensional forms in Fig. 3 and they are not chosen arbitrarily. A set of properties needs to be satisfied so that ϕ is uniquely determined. Among those properties, we have: (i) zeroth moment condition $\sum_{\mathbf{x}_{i,j} \in \mathcal{D}_{\mathbf{X}_n}} \phi(r_x) = 1$, (ii) first moment condition $\sum_{\mathbf{x}_{i,j} \in \mathcal{D}_{\mathbf{X}_n}} (x_{i,j} - X_n) \phi(r_x) = 0$, (iii) second moment condition $\sum_{\mathbf{x}_{i,j} \in \mathcal{D}_{\mathbf{X}_n}} (x_{i,j} - X_n)^2 \phi(r_x) = K$, with K a constant²⁶, (iv) third moment condition $\sum_{\mathbf{x}_{i,j} \in \mathcal{D}_{\mathbf{X}_n}} (x_{i,j} - X_n)^3 \phi(r_x) = 0$, and (v) sum of squares condition $\sum_{\mathbf{x}_{i,j} \in \mathcal{D}_{\mathbf{X}_n}} (\phi(r_x))^2 = C$, with C a constant²⁶.

All mentioned kernel functions ϕ and their corresponding moment conditions are presented

Article

TABLE I: Moment conditions for all continuous kernel functions ϕ .

	Zeroth Moment (i)	First Moment (ii)	Second Moment (iii)	Third Moment(iv)	Sum of Squares Condition (v)	Order of Accuracy
ϕ_1	✓	✓	×	×	✓ ($C = 0.5$)	2^{nd}
ϕ_2	✓	✓	✓ ($K = \frac{38}{60} - \frac{\sqrt{69}}{60}$)	✓	✓ ($C = 0.3925$)	4^{th}

in Table I. Since conditions (i) and (ii) together ensure that a smoothed function is interpolated with second-order accuracy, consequently, ϕ_1 is second-order accurate. A fourth-order accuracy is manifested by ϕ_2 since it satisfies the third moment condition (iv). Both kernel functions ϕ_1 and ϕ_2 satisfy the sum of squares condition (v), where each value of C is mentioned in Table I. Note that all kernel function properties and conditions are summarized in Ref. 26.

D. 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)^{8,14}, Finite Volume Method (FVM)^{18,28}, Finite Element Method (FEM)^{29,30}, and Lattice Boltzmann Method (LBM)^{15,31}. These DF-IBM algorithms are inserted inside the incompressible Navier-Stokes equations where the $p - \mathbf{U}$ coupling is handled by using either the projection method (fractional step method) and its variants as in Refs. 8, 14, and 18, the SIMPLE algorithm³², and the PISO algorithm^{33,34}. Regardless, of the numerical technique and $p - \mathbf{U}$ decoupling algorithm used, the DF-IBM steps present in every approach, are immutable. The generic steps are ordered as follows:

1. Eulerian Velocity Interpolation Step:

$$\mathbf{U}(\mathbf{X}_n, t) = \mathcal{I}[\mathbf{u}^*(\mathbf{x}_{i,j}, t)]. \quad (19)$$

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 $\mathbf{X}_n$.

Article

2. Direct-Forcing Step:

$$\mathbf{F}_{IB}(\mathbf{X}_n, t) = \frac{\mathbf{U}_{IB}(\mathbf{X}_n, t) - \mathbf{U}(\mathbf{X}_n, t)}{\Delta t}, \quad (20)$$

where $\mathbf{U}_{IB}$ represents the desired (or target) 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 Step:

$$\mathbf{f}(\mathbf{x}_{i,j}, t) = \mathcal{S}[\mathbf{F}_{IB}(\mathbf{X}_n, t)]. \quad (21)$$

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. Update Eulerian Velocity Step:

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:

$$\tilde{\mathbf{u}}(\mathbf{x}_{i,j}, t) = \mathbf{u}^*(\mathbf{x}_{i,j}, t) + \Delta t \mathbf{f}(\mathbf{x}_{i,j}, t). \quad (22)$$

1. Immersed boundary no-slip velocity boundary condition

The spreading and the interpolation operators in the DF-IBM are reciprocal, i.e., inversely related. Once this reciprocity is violated, a no-slip boundary error is generated, allowing the fluid to penetrate the solid domain.

This boundary error problem can be manifested by using the first explicit DF-IBM approach proposed by Uhlmann⁸, and employing the concept of multi-DF-IBM found in Refs. 9–11 that will help understand the roots of this error and how it can be resolved.

The error will be explained in a simplified context, assuming a Lagrangian domain represented by one marker ($\mathbf{X}_1$) and surrounded by two equidistant Eulerian grid points ($\mathbf{x}_1, \mathbf{x}_2$), and assuming

Article

that $\delta_1 = 0.6$ and $\delta_2 = 0.4$ while keeping in mind the zeroth moment condition. If one proceeds with the explicit DF-IBM, i.e., one iteration as in Ref. 8 and assumes that the Lagrangian weight is almost equal to the Eulerian weight, as proposed by Uhlmann, with $W_n = \Delta V_{i,j} = 1$ for simplicity, the following equations for the interpolation, direct-forcing, spreading, and updated Eulerian velocity steps for a single time-step can be obtained:

$$\mathbf{U}(\mathbf{X}_1, t) = \mathbf{u}^*(\mathbf{x}_1, t)\delta_1 + \mathbf{u}^*(\mathbf{x}_2, t)\delta_2,$$

$$\mathbf{F}_{IB}(\mathbf{X}_1, t) = \frac{\mathbf{U}_{IB}(\mathbf{X}_1, t) - \mathbf{U}(\mathbf{X}_1, t)}{\Delta t},$$

$$\mathbf{f}(\mathbf{x}_1, t) = \mathbf{F}_{IB}(\mathbf{X}_1, t)\delta_1 \quad \& \quad \mathbf{f}(\mathbf{x}_2, t) = \mathbf{F}_{IB}(\mathbf{X}_1, t)\delta_2,$$

$$\tilde{\mathbf{u}}(\mathbf{x}_1, t) = \mathbf{u}^*(\mathbf{x}_1, t) + \Delta t \mathbf{f}(\mathbf{x}_1, t) \quad \& \quad \tilde{\mathbf{u}}(\mathbf{x}_2, t) = \mathbf{u}^*(\mathbf{x}_2, t) + \Delta t \mathbf{f}(\mathbf{x}_2, t).$$

The no-slip boundary error disappears once $\mathbf{U}_{IB}(\mathbf{X}_1, t) = \mathcal{J}[\tilde{\mathbf{u}}]$, which means that the no-slip boundary condition is exactly enforced on the Lagrangian Marker $\mathbf{X}_1$. However, this is not the case since:

$$\mathcal{J}[\tilde{\mathbf{u}}] = \mathbf{U}_{IB}(\mathbf{X}_1, t) \left(\delta_1^2 + \delta_2^2 \right) + \mathbf{U}(\mathbf{X}_1, t) \left(1 - \left(\delta_1^2 + \delta_2^2 \right) \right), \quad (24)$$

with $(\delta_1 + \delta_2) = 1$, according to the zeroth moment condition, while $(\delta_1^2 + \delta_2^2) = C \neq 1$ according to the sum of squares condition described earlier.

Now employing the concept of multiple DF-IBM iterations, that leads to the family of explicit or implicit IBMs, depending on whether a fixed number of iterations is already prescribed or if a tolerance between two successive DF-IBM iterations is defined to satisfy the no-slip boundary condition. After m iterations, the interpolated Eulerian velocity field is obtained as follows:

$$\mathcal{J}[\tilde{\mathbf{u}}^{(m)}] = \mathbf{U}_{IB}(\mathbf{X}_1, t) \mathcal{A} + \mathbf{U}^{(0)}(\mathbf{X}_1, t) \mathcal{B}, \quad (25)$$

where $\mathcal{A} = \sum_{i=0}^{m-1} C(1-C)^i$, and $\mathcal{B} = (1-C)^m$.

The superscript (m) represents the number of iterations, having $\tilde{\mathbf{u}}^{(0)} = \mathbf{u}^*$ and $\mathbf{U}^{(0)}(\mathbf{X}_1, t) = \mathcal{J}[\mathbf{u}^*]$. The ideal case is to have the term $\mathcal{A} = 1$ and the term $\mathcal{B} = 0$, to satisfy the no-slip boundary condition $\mathbf{U}_{IB}(\mathbf{X}_1, t) = \mathcal{J}[\tilde{\mathbf{u}}^{(m)}] = \mathbf{U}^{(m)}(\mathbf{X}_1, t)$ and impose the reciprocity on the interpolation and spreading operations. Table II reveals how the terms $\mathcal{A}$ and $\mathcal{B}$ behave when the multi-DF is

Article

TABLE II: Multi-DF-IBM effect on the interpolated Eulerian velocity after m iterations.

m	Term $\mathcal{A}$	Term $\mathcal{B}$
0	0.0	1.0
1	0.52	0.48
2	0.7696	0.2304
20	0.99999957	0.00000043
∞	1.0	0.0

implemented. Clearly, the term $\mathcal{A} \rightarrow 1$ and the term $\mathcal{B} \rightarrow 0$ when the number of iterations m increases.

The no-slip boundary error is directly related to the Lagrangian weight W_n , used in the force spreading operator. According to Uhlmann⁸, the Lagrangian weight was firstly set equal to the Eulerian weight, which is equal in a $2D$ Cartesian uniform mesh, to h^2 . This method was found to be inaccurate¹⁷, due to the Lagrangian representation of the immersed boundary having a dimensional issue since it is described only through its boundary and not the whole volume, so one dimension is missing 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 Ref. 17, and therefore should not be equal. Different methods to calibrate the Lagrangian weight computations can be found in the literature that improve the mathematical reciprocity of the interpolation/spreading operators. The methods used in this research are described in the next section of this paper.

Besides the DF-IBM iterations, another approach to reduce the no-slip boundary error is by decreasing the simulation time-step, as stated in Refs. 10, 11, and 16.

2. Lagrangian weight computation methods

The Lagrangian weight was primarily set equal to the Eulerian weight, as adopted by the majority of researchers^{8,10,14,35}:

$$W_n = \Delta V_{i,j}. \quad (26)$$

Another method was proposed by Pinelli et al.¹⁸ to enforce the reciprocity on the interpolation and spreading operators, where the Lagrangian weight was computed by solving the following

Article

system of linear equations:

$$\mathbf{A}\mathbf{W}_n = \mathbf{1}, \quad (27)$$

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

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

The system is solved by using the Krylov subspace solver, BiCGStab. The advantage of this method is that it can be used on uniform or non-uniform Eulerian grids and it reduces the no-slip boundary error¹⁷, while increasing the computational cost caused by the iterative solving of the system of linear equations, having the computational cost directly proportional to the total number of Lagrangian markers N_{Lag} .

Furthermore, an analytical correction for the Lagrangian weight was proposed in Refs. 15 and 16, 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 performed on the Lagrangian weight was deduced from Eq. (24), i.e., W_n is divided by the constant C obtained from the sum of squares condition imposed by the kernel function used, meaning that this correction is kernel function dependent since the sum of squares condition is not satisfied by all kernel functions, hence can be applied for specific set of kernel functions as shown in Table I. If we re-calculate Eq. (24) using the described correction, we satisfy exactly the no-slip boundary condition for this particular case. For a general case where multiple Lagrangian markers exist enclosed each by multiple Eulerian grid points, the condition is not purely satisfied. However, it was shown in Refs. 15 and 16 that it could still eliminate some of the 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.¹⁷:

$$W_n = \frac{1}{\sum_{\mathbf{x}_{i,j} \in \mathcal{D}_{\mathbf{X}_m}} \sum_{\mathbf{X}_n \in \mathcal{D}_{\mathbf{x}_{i,j}}} \delta_h(\mathbf{x}_{i,j} - \mathbf{X}_n) \delta_h(\mathbf{x}_{i,j} - \mathbf{X}_m) h^2}. \quad (28)$$

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

Article

function and its computational cost is considerably lower compared to the method developed by Pinelli et al.¹⁸. The difference between Eq. (27) and (28) for the no-slip boundary error was shown to be minimal in Ref. 17.

IV. CURRENT DF-IBM ALGORITHM

We now propose an efficient implicit DF-IBM variant that reduces the computational effort of previous implicit IBMs, such as the IBPM of Taira & Colonius¹² and the IBM algorithm of Ji et al.¹³, where the system's dual constraints problem formed for an incompressible flow problem, are fulfilled sequentially in the solution process, using the PISO algorithm as the Navier-Stokes fluid solver.

Modifications to the original PISO algorithm are presented here, for the velocity field to satisfy the two constraints problem in the predictor-corrector algorithm formed by:

- (a) No-slip velocity boundary condition on the Lagrangian markers, $\mathbf{U}_{IB}^{n+1} = \mathcal{J}[\tilde{\mathbf{u}}]$ on Γ_s ,
- (b) Divergence-free condition on Eulerian grid points, $\nabla \cdot \mathbf{u}^{n+1} = 0$ in Ω_f .

In addition, a stability criterion needs to be fulfilled for the PISO algorithm, stating that the courant (CFL) number has to be limited to less than one. The two constraints are decoupled from the Navier-Stokes momentum equations through the operator-splitting of the PISO algorithm. The following procedure is adopted at each time-step, having $\mathbf{u}$, p , and $\mathbf{f}$ solved sequentially, and up until reaching convergence, where the superscript n and $n + 1$ represent the past (known) and current time (unknown) levels, respectively:

1. Momentum Predictor:

An initial implicit prediction of the velocity field $\mathbf{u}^*$ in the Eulerian grid is calculated using Eq. (4), without the addition of the boundary force term, while the pressure field p from previous time-step t^n is used for the calculation:

$$a_P \mathbf{u}_P^* + \sum_N a_N \mathbf{u}_N^* = \mathbf{RHS}(\mathbf{u}^n, \mathbf{u}^{n-1}) - (\nabla p)^n. \quad (29)$$

We note that the predicted velocity does not satisfy the divergence-free condition, $\nabla \cdot \mathbf{u}^* \neq 0$.

2. DF-IBM Iterations:

Inner iterations start for $m = 1$, having $\tilde{\mathbf{u}}^{(0)}(\mathbf{x}_{i,j}, t^{n+1}) = \mathbf{u}^*$ and $\mathbf{F}_{IB}^{(0)}(\mathbf{X}_n, t^{n+1}) = \mathbf{F}_{IB}(\mathbf{X}_n, t^n)$.

The iterations procedure is carried out using the following steps:

Article

(a) Eulerian Velocity Interpolation:

$$\mathbf{U}^{(m-1)}(\mathbf{X}_n, t^{n+1}) = \mathcal{I}[\tilde{\mathbf{u}}^{(m-1)}(\mathbf{x}_{i,j}, t^{n+1})], \quad (30)$$

(b) Direct-Forcing:

$$\mathbf{F}_{IB}^{(m)}(\mathbf{X}_n, t^{n+1}) = \frac{\mathbf{U}_{IB}(\mathbf{X}_n, t^{n+1}) - \mathbf{U}^{(m-1)}(\mathbf{X}_n, t^{n+1})}{\Delta t} + \mathbf{F}_{IB}^{(m-1)}(\mathbf{X}_n, t^{n+1}), \quad (31)$$

(c) Lagrangian Force Spreading:

$$\mathbf{f}^{(m)}(\mathbf{x}_{i,j}, t^{n+1}) = \mathcal{I}[\mathbf{F}_{IB}^{(m)}(\mathbf{X}_n, t^{n+1})], \quad (32)$$

(d) Update Eulerian Velocity:

$$a_P \tilde{\mathbf{u}}_P^{(m)} + \sum_N a_N \tilde{\mathbf{u}}_N^{(m)} = \mathbf{RHS}(\mathbf{u}^n, \mathbf{u}^{n-1}) - (\nabla p)^n + \mathbf{f}^{(m)}, \quad (33)$$

(e) The no-slip velocity boundary condition requirement:

$$\mathbf{U}_{IB}(\mathbf{X}_n, t^{n+1}) = \mathcal{I}[\tilde{\mathbf{u}}^{(m)}(\mathbf{x}_{i,j}, t^{n+1})] = \mathbf{U}^{(m)}(\mathbf{X}_n, t^{n+1}). \quad (34)$$

A convergence criterion is set for two consecutive DF-IBM iterations to check whether the updated predicted velocity field satisfies the no-slip boundary condition on the Lagrangian markers:

$$\|E_{\tilde{\mathbf{u}}^{(m)}}\|_2 - \|E_{\tilde{\mathbf{u}}^{(m-1)}}\|_2 < \varepsilon, \quad (35)$$

with

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

and

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

where ε is a user-specified tolerance, based on whether an accurate no-slip boundary condition is required.

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

$$\mathbf{f}^{n+1} = \mathbf{f}^{(m)}(\mathbf{x}_{i,j}, t^{n+1}) \quad \& \quad \tilde{\mathbf{u}} = \tilde{\mathbf{u}}^{(m)}(\mathbf{x}_{i,j}, t^{n+1}), \quad (36)$$

otherwise, return to Step 2 (a), and let $m = m + 1$.

It can be noted, that at this stage, the updated predicted velocity still does not satisfy the divergence-free condition, $\nabla \cdot \tilde{\mathbf{u}} \neq 0$, while the no-slip boundary condition is fulfilled $\mathbf{U}^{(m)}(\mathbf{X}_n, t^{n+1}) = \mathcal{I}[\tilde{\mathbf{u}}^{(m)}(\mathbf{x}_{i,j}, t^{n+1})] = \mathbf{U}_{IB}(\mathbf{X}_n, t^{n+1})$.

Article

3. PISO Corrector Loops:

Inner corrector loops start for $m = 1$, where $1 \leq m \leq m_f$, with $m_f = 3$ is the fixed number of loops. For $m = 1$ we have, $\hat{\mathbf{u}}^{(0)} = \tilde{\mathbf{u}}$:

- (a) To enforce the divergence-free constraint, the mPPE represented by Eq. (10a) is used here to compute the new pressure field:

$$\nabla \cdot \left(\frac{1}{a_p} (\nabla p)^{(m)} \right) = \nabla \cdot \left(\frac{\mathbf{H}(\hat{\mathbf{u}}^{(m-1)}, \mathbf{u}^n, \mathbf{u}^{n-1})}{a_p} \right) + \nabla \cdot \left(\frac{1}{a_p} \mathbf{f}^{n+1} \right), \quad (37)$$

- (b) The velocity field is then corrected explicitly from the former pressure solution $p^{(m)}$ using Eq. (12):

$$\hat{\mathbf{u}}^{(m)} = \frac{\mathbf{H}(\hat{\mathbf{u}}^{(m-1)}, \mathbf{u}^n, \mathbf{u}^{n-1})}{a_p} - \frac{1}{a_p} (\nabla p)^{(m)} + \frac{1}{a_p} \mathbf{f}^{n+1}, \quad (38)$$

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

$$\mathbf{u}^{n+1} = \hat{\mathbf{u}}^{(m_f)}, \quad (39)$$

$$p^{n+1} = p^{(m_f)}. \quad (40)$$

Note that this new velocity field satisfies the divergence-free constraint $\nabla \cdot \mathbf{u}^{n+1} = 0$.

A flowchart of the current modified PISO algorithm with the DF-IBM is described in Fig. 4, for one simulation time-step.

The proposed algorithm is based on the original PISO method of Issa²⁴, which has the velocity, pressure, and boundary force decoupled, following the operator-splitting strategy of the PISO algorithm, to enforce the no-slip boundary condition on the Lagrangian domain and the divergence-free condition on the Eulerian domain in a segregated manner. Although the no-slip boundary constraint is satisfied before the corrector loops of the PISO algorithm, the Lagrangian interpolated velocity remains primarily the same even if computed after the corrector loops, in accordance with previous findings^{10,14}. Similar to the work of Ji et al.¹³, the mPPE differs from the PPE found in IPBM¹² by having the boundary force already satisfying the no-slip boundary condition. In our case, the divergence-free constraint is only fulfilled by the mPPE allowing the use of fast iterative solvers such as GAMG or Krylov subspace CG solvers without any modification. This avoids changing the coefficient matrix and undermining the condition number of the PPE, which leads to convergence problems similar to those encountered by the IPBM¹².

Article

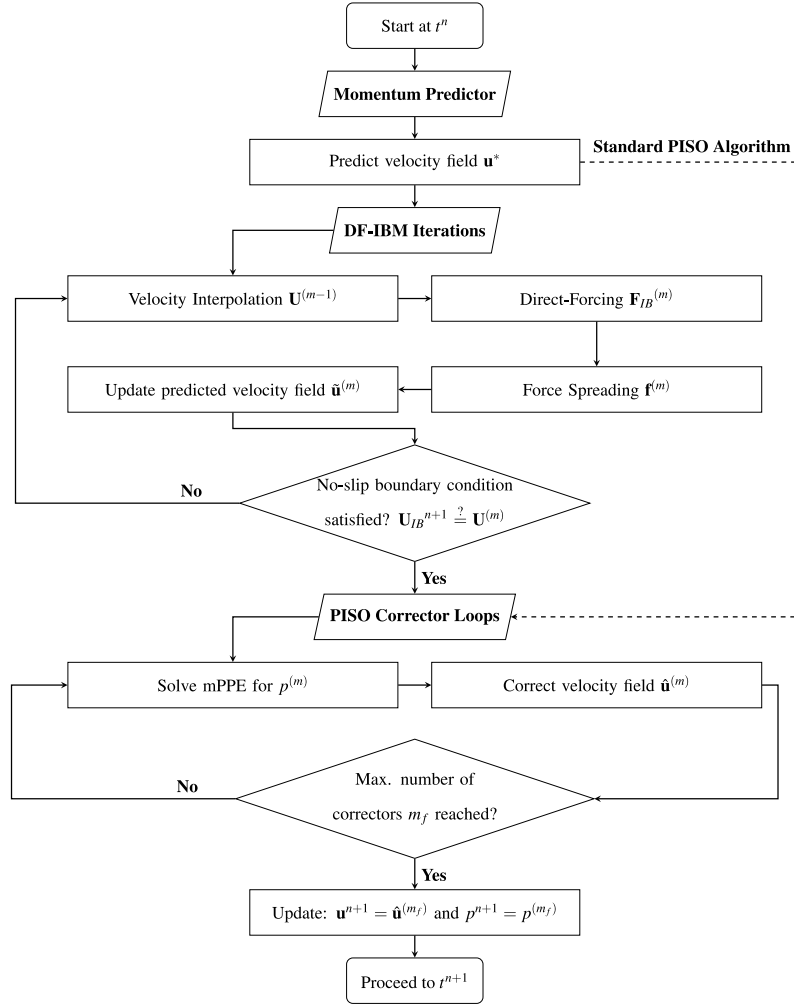


FIG. 4: Flowchart of the modified PISO algorithm with the current DF-IBM for one simulation time-step.

An additional attractive aspect of this current algorithm is the rapid process to achieve a no-slip boundary condition on the Lagrangian domain. This is done by replacing Eq. (26) of Uhlmann⁸ for computing the Lagrangian weight with Eq. (27) from Pinelli et al.¹⁸ for stationary immersed boundary problems and Eq. (28) found in Ref. 17 for moving immersed boundaries. In the lat-

Article

ter case, since the Lagrangian weight is re-computed at each time level, to avoid increasing the computational effort by solving the linear system in Eq. (27) at each time-step, the analytical Lagrangian weight computation expressed by Eq. (28) is used instead. It was determined through a trial and error process, that an additional correction of $3/2$ to Eqs. (27) and (28) for the Lagrangian weight computation, i.e., by simply multiplying the Lagrangian weight by this factor, was notably speeding up the no-slip boundary convergence while keeping the forces on the immersed boundary intact. Furthermore, the Lagrangian boundary force initialization method for the term $\mathbf{F}_{IB}^{(m-1)}$ used in Eq. (31) was also affecting the efficiency of the no-slip boundary convergence. It is therefore proposed here to initialize $\mathbf{F}_{IB}^{(0)}$ at the beginning of a new time level by the value from the previous time level, i.e., $\mathbf{F}_{IB}^{(0)} = \mathbf{F}_{IB}''$ instead of initializing by zero. Finally, a convergence criterion for the no-slip boundary condition was set for two consecutive DF-IBM iterations, since it was reported in Ref. 9, that after multiple iterations the error convergence rate becomes very minimal, hence iterating even further will not affect the desired solution. This significantly reduces the number of iterations per time-step.

The proposed DF-IBM variant is distinguished by its ease of understanding and implementation. The current algorithm can be readily integrated into any incompressible fluid solver that utilizes the PISO algorithm or other decoupling algorithms, without significantly increasing computational memory requirements. Only minimal modifications to the original system equations are necessary, specifically in the momentum equations and the PPE.

A. OpenFOAM implementation

The native transient incompressible OpenFOAM solver, `pisoFoam`, has been modified for this research. The original predictor step Eq. (29) of the PISO solver is kept unchanged. The function `createIBMLagSupport()` is called afterward for the Lagrangian support domain creation for each Lagrangian marker. This function is called once for a stationary immersed boundary, and called for each time-step when moving cases are considered. Once the support domains are created, the modified solver checks whether a prescribed motion case is of interest. If this is the case, then the function `movingIBM()` is called. This function computes solid body kinematics, i.e., the linear and angular position and velocity depending on the prescribed equations of motion. The DF-IBM iterations are initiated through the `calculationIBM()` function, which applies the interpolation Eq. (30), direct-forcing Eq. (31), and spreading Eq. (32) steps. To update the Eulerian

Article

velocity as in Eq. (33) of the current algorithm, the virtual function `solve` of OpenFOAM is used with the addition of the boundary force which has the following data type `volVectorField`. The function `noSlipErrorVerificationCheck()` is called next, to check if the no-slip velocity boundary constraint is satisfied. The corrector loop of the original `pisoFoam` is also modified to account for the presence of the solid immersed boundary. The additional term in the mPPE Eq. (37) is of the type `surfaceScalarField`, where the divergence is represented by the explicit differential operator `fvc::div()`, which is added to the original PPE of the `pisoFoam` solver. The boundary force is finally added to the explicit velocity corrector step of the native PISO solver, now represented by Eq. (38).

V. RESULTS

Simulations were performed on the HPC PILCAM2 platform of UTC, built with a CPU AMD EPYC 7552 2.2GHz 192MB L3 Cache. The number of cores varied from 12 to 18, depending on the mesh size and the Reynolds number.

A. Taylor–Green decaying vortices: Accuracy study

To validate the temporal and spatial accuracy of the proposed discretization in the present method, we consider the unsteady Taylor–Green decaying vortices problem in two-dimensional form. The exact analytical solution of this test problem is given by:

$$u(x, y, t) = -u_0 \cos\left(\frac{\pi}{L}x\right) \sin\left(\frac{\pi}{L}y\right) e^{-2\left(\frac{\pi}{L}\right)^2 \nu t}, \quad (41a)$$

$$v(x, y, t) = u_0 \sin\left(\frac{\pi}{L}x\right) \cos\left(\frac{\pi}{L}y\right) e^{-2\left(\frac{\pi}{L}\right)^2 \nu t}, \quad (41b)$$

$$p(x, y, t) = p_0 - \frac{u_0^2}{4} \left[\cos\left(\frac{2\pi}{L}x\right) + \cos\left(\frac{2\pi}{L}y\right) \right] e^{-4\left(\frac{\pi}{L}\right)^2 \nu t}. \quad (41c)$$

In this analysis, the simulation is conducted in a quadrilateral domain $\Omega_f = [-\pi, \pi] \times [-\pi, \pi]$ m, having a circular cylinder of unitary radius immersed inside the squared domain and located at its center (0,0), as shown in Fig. 5. The initial boundary conditions at $t = 0$ and the time-dependent boundary conditions for the whole Eulerian domain for the velocity and pressure are imposed by Eqs. (41a) to (41c), and the time-dependent boundary condition, i.e., the desired

Article

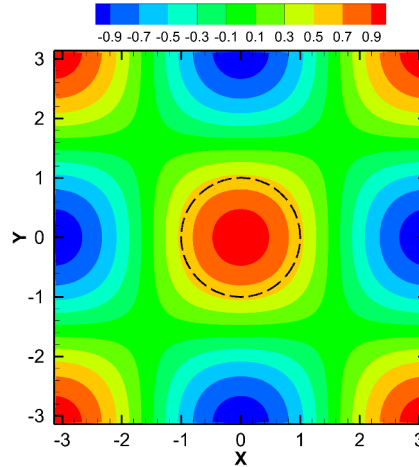


FIG. 5: Eulerian domain, vorticity contours, and the cylinder immersed boundary at $t = 0.34s$.

Lagrangian velocities at the cylinder immersed boundary, are imposed by Eqs. (41a) and (41b). The simulation is advanced to a final time of $0.34 s$, close to the half-life of the decaying vortices. Periodic boundary conditions are imposed on the Eulerian domain's outer boundaries, p_0 is set to zero, and the $Re = u_0 L / \nu$ is set to π , where $u_0 = 1 m/s$ is the initial maximum velocity, $L = \pi m$ is the vortex size, and $\nu = 1 m^2/s$ is the kinematic viscosity. The 3-point-width function ϕ_1 is selected.

For the temporal accuracy study, simulations were carried out for different values of the time-step $\Delta t = 0.02, 0.01, 0.005, 0.0025, 0.00125 s$, having the numerical results compared to a refined solution obtained for a very small time-step of $\Delta t = 0.0001 s$. A fixed Eulerian grid of 320×320 was set for all simulations to cancel out the error due to the spatial discretization. The errors of the u -velocity component in the x direction are computed using L_1 , L_2 , and L_∞ norms as well as the error convergence rate represented by Eqs. (A1) to (A4), respectively found in Appendix (A). Table III shows that the convergence rate of the temporal error of the current method is about 2 for all norms, as expected, meaning that the error is decreased by a factor of 4 for all norms when the time-step is divided by 2. The same results are plotted in a log-log plot in Fig. 6a, and it is worth noting that the same behavior was observed for the v -velocity component confirming the second-order temporal accuracy of the present method.

For the spatial accuracy study, two separate analyses were carried out, one without the im-

Article

TABLE III: Temporal error convergence analysis for the u -velocity component of the current solver.

Δt (s)	$\ E_u\ _1$	Order	$\ E_u\ _2$	Order	$\ E_u\ _\infty$	Order
0.02	2.01×10^{-4}		2.48×10^{-4}		4.92×10^{-4}	
0.01	4.93×10^{-5}	2.03	6.06×10^{-5}	2.03	1.20×10^{-4}	2.02
0.005	1.20×10^{-5}	2.03	1.48×10^{-5}	2.02	3.01×10^{-5}	2.00
0.0025	3.00×10^{-6}	2.00	3.70×10^{-6}	2.00	8.06×10^{-6}	1.90
0.00125	7.68×10^{-7}	1.96	9.92×10^{-7}	1.90	2.45×10^{-6}	1.71

mersed boundary, and another with the inclusion of the immersed boundary since it is known that the diffused DF-IBM deteriorates the overall spatial accuracy of the method due to the discrete Dirac delta function. The analytical solution was compared to numerical solutions for different Eulerian grid sizes $N_x \times N_y = 20 \times 20, 40 \times 40, 80 \times 80, 160 \times 160$, and 320×320 . A fine time-step $\Delta t = 0.0001$ s was used to ensure that the temporal error is negligible compared to the spatial error. The first analysis without the immersed cylinder is carried out, and the results obtained for the convergence rate of L_1 , L_2 , and L_∞ norms for the error of the u -velocity component are written in Table IV. A second-order accuracy was recorded for the spatial discretization without the immersed boundary for all norms as expected since the second-order accurate central differencing interpolation scheme was used for the convection term in this case. The second analysis includes the immersed cylinder where the number of Lagrangian markers representing Γ_s are $N_{Lag} = 20, 40, 80, 160$, and 320 for all Eulerian grid sizes respectively. The results are presented in Table V, where a similar second-order error convergence rate is detected which confirms the accuracy of the discrete Dirac delta function in the case of smooth fields as reported in Ref. 8. Fig 6b shows the error of the u -velocity component function of the grid size in a log-log plot for both cases with and without the immersed boundary. It is evident that the immersed boundary introduces error in the original discretization, but it maintains the same order of accuracy for smooth fields as in the case of the Taylor-Green problem.

Article

TABLE IV: Spatial error convergence analysis for the u -velocity component of the current solver without an immersed boundary.

$N_x \times N_y$	$\ E_u\ _1$	Order	$\ E_u\ _2$	Order	$\ E_u\ _\infty$	Order
20×20	1.12×10^{-3}		1.39×10^{-3}		2.80×10^{-3}	
40×40	2.84×10^{-4}	1.98	3.50×10^{-4}	1.98	7.00×10^{-4}	2.00
80×80	7.16×10^{-5}	1.99	8.83×10^{-5}	1.99	1.76×10^{-4}	1.98
160×160	1.79×10^{-5}	1.99	2.21×10^{-5}	1.99	4.43×10^{-5}	1.99
320×320	4.48×10^{-6}	2.00	5.53×10^{-6}	2.00	1.11×10^{-5}	1.98

TABLE V: Spatial error convergence analysis for the u -velocity component of the current solver with an immersed boundary.

$N_x \times N_y$	$\ E_u\ _1$	Order	$\ E_u\ _2$	Order	$\ E_u\ _\infty$	Order
20×20	2.51×10^{-3}		3.71×10^{-3}		1.18×10^{-2}	
40×40	6.23×10^{-4}	2.01	9.18×10^{-4}	2.01	3.05×10^{-3}	1.95
80×80	1.52×10^{-4}	2.02	2.24×10^{-4}	2.03	7.71×10^{-4}	1.98
160×160	3.77×10^{-5}	2.01	5.54×10^{-5}	2.01	2.08×10^{-4}	1.88
320×320	9.39×10^{-6}	2.00	1.37×10^{-5}	2.00	5.43×10^{-5}	1.94

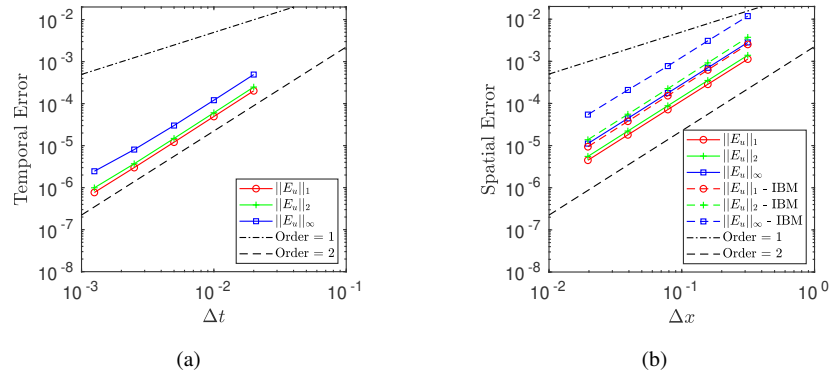


FIG. 6: Temporal (a) and spatial (b) accuracy study of the Taylor-Green decaying vortices for the present solver.

Article

B. DF-IBM strategies: Performance analysis

The performance and efficiency of different DF-IBM strategies are assessed quantitatively and qualitatively against the present strategy where the PISO decoupling algorithm is dropped out for this study, to examine the different DF-IBM sub-routine's performance alone for one single execution. The performance is analyzed through the following test as in Ref. 14:

1. Initialise velocity field in x-direction:

$$u_0(x, y) = e^x \cos(y) - 2 \quad 0 \leq (x, y) \leq 1 \quad \text{in } \Omega_f,$$

2. Find f such that:

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

The following test is conducted over a circular cylinder of radius $0.2 \, m$ with its center located at $(0.52, 0.54)$ in an Eulerian squared domain $\Omega_f = [0, 1] \, m \times [0, 1] \, m$. The cylinder is considered stationary, therefore $U_{IB} = 0$. The domain is covered by 64×64 Eulerian grid points in the streamwise (x) and transverse (y) directions respectively, having $h = 0.015625 \, m$ and the cylinder boundary Γ_s is represented by 80 Lagrangian markers where $\Delta s \approx 0.015708 \, m$, Δt is fixed to $1 \, s$ and ϕ_1 is the kernel function used. Slip boundary conditions are applied to all domain boundaries.

Because the desired Lagrangian velocity U_{IB} is set to zero for all Lagrangian markers, the no-slip boundary error on the immersed boundary is defined as the computed non-zero velocity at Lagrangian markers. For this analysis only, a stricter convergence criterion is set unlike the one specified in Eq. (35) for two successive DF-IBM iterations, which reads:

$$\begin{aligned} \|E_{no-slip}\|_2 &= \|E_{\tilde{u}^{(m)}}\|_2 \\ &= \left\| U_{IB}(X_n, Y_n) - \mathcal{J}[\tilde{u}^{(m)}(x_{i,j}, y_{i,j})] \right\|_2 < \varepsilon, \end{aligned} \quad (42)$$

since we are interested in the updated no-slip boundary error through the newly updated velocity field and not the error between two successive DF-IBM iterations.

Table VI compares the number of iterations to reach the no-slip boundary constraint, the CPU time to satisfy this constraint, and the no-slip boundary error using L_1 , L_2 , and L_∞ norms. The tolerance was set to 1×10^{-15} for the implicit DF-IBM strategies using the convergence criterion defined by Eq. (42).

Article

TABLE VI: Performance comparison of different DF-IBM variants in terms of iterations, CPU time, and no-slip boundary error.

DF-IBM	Explicit/Implicit	Lagrangian Weight	Iterations	CPU Time	$\ E_{no-slip}\ _1$	$\ E_{no-slip}\ _2$	$\ E_{no-slip}\ _\infty$
Variant	Strategy	Method	(m)	(s)			
Uhlmann ⁸	Explicit	Eq. (26)	1	1.20×10^{-3}	2.77×10^{-1}	3.02×10^{-1}	4.36×10^{-1}
Pinelli et al. ¹⁸	Explicit	Eq. (27)	1	1.21×10^{-3}	4.88×10^{-4}	5.58×10^{-4}	1.11×10^{-3}
Cheyman et al. ¹⁷	Explicit	Eq. (28)	1	1.17×10^{-3}	2.83×10^{-3}	4.63×10^{-3}	1.65×10^{-2}
Multi-DF-IBM ⁹⁻¹¹	Explicit	Eq. (26)	3	1.70×10^{-3}	6.91×10^{-2}	7.55×10^{-2}	1.17×10^{-1}
Ji et al. ¹³	Implicit	Eq. (26)	807	1.10×10^{-1}	6.03×10^{-16}	9.98×10^{-16}	2.87×10^{-15}
Present	Implicit	Eq. (27)	415	6.48×10^{-2}	5.57×10^{-16}	9.78×10^{-16}	3.19×10^{-15}
Present	Implicit	Eq. (28)	402	6.16×10^{-2}	5.52×10^{-16}	9.59×10^{-16}	3.25×10^{-15}
Present	Implicit	Eq. (27) + Correction	273	4.46×10^{-2}	5.23×10^{-16}	9.18×10^{-16}	3.10×10^{-15}
Present	Implicit	Eq. (28) + Correction	264	4.26×10^{-2}	5.40×10^{-16}	9.36×10^{-16}	3.20×10^{-15}

The explicit DF-IBM of Uhlmann⁸, Pinelli et al.¹⁸, and Cheylan et al.¹⁷, all required almost the same CPU time since only one iteration is needed due to their explicit nature, while keeping in mind that the time needed to compute the Lagrangian weight was not included. The Lagrangian weight method using Eq. (27) reduced the error for the L_2 norm with respect to Uhlmann's result by a factor of 541 while using Eq. (28) reduced the error by a factor of 65. This behavior is expected since these two methods to compute the Lagrangian weight improve the reciprocity of the interpolation and spreading operators of the DF-IBM. The multi-DF-IBM of Luo et al.⁹, Kempe & Fröhlich¹⁰, and Breugem¹¹ apply 3 iterations in their algorithms using the Lagrangian weight method suggested by Uhlmann⁸ i.e., Eq. (26). An error reduction by a factor of 4 was recorded for the L_2 norm. Nevertheless, all of these errors obtained using the explicit approach are still considered large.

The implicit strategy proposed by Ji et al.¹³ required a total of 807 iterations to drive the error to the prescribed tolerance, where the CPU time increased almost by a factor of 91 compared to Uhlmann's variant. On the other hand, the present implicit DF-IBM variant required 415 and 402 iterations when using Eqs. (27) and (28) for the computation of the Lagrangian weight respectively, having the CPU time increased by a factor of 54 and 51 compared to Uhlmann's. The analytical correction proposed here, increased the CPU time only by a factor of 37 and 35 when using Eqs. (27) and (28) for the Lagrangian weight respectively, showing a decrease of almost 60% in

Article

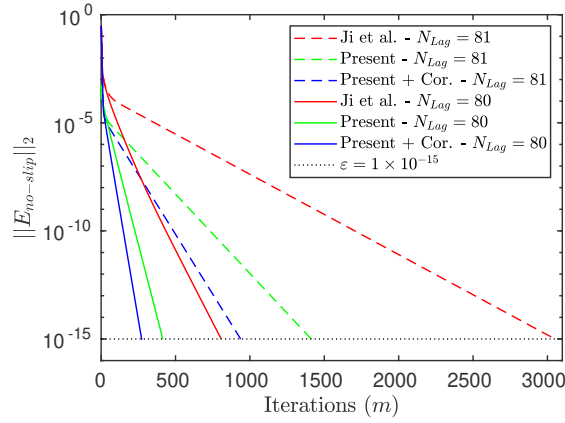


FIG. 7: No-slip boundary error against the number of iterations for odd and even number of Lagrangian markers. The results of the present DF-IBM with Eq. (27) for the Lagrangian weight method are used for comparison.

CPU time with the present implicit DF-IBM when using Lagrangian weight methods that enhance the reciprocity of the linear operators along with the correction proposed, compared to the implicit variant of Ji et al.¹³.

It is worth noting, that defining the Lagrangian immersed boundary Γ_s using an odd number of Lagrangian markers drastically increased the number of iterations, see Ref. 14. If 81 Lagrangian markers were used to represent the circular cylinder, a total of 3051 iterations would be needed for the implicit method of Ji et al.¹³, whereas for the present DF-IBM 1413 iterations would be required to reach the imposed tolerance using Eq. (27) for the Lagrangian weight when no correction is included, and 938 iterations when the correction is applied, as shown in Fig. 7. Therefore, it is always recommended by the authors to use an even number for the immersed boundary Lagrangian representation.

All these results are reflected qualitatively in the velocity contours over the Eulerian domain in Fig. 8. Uhlmann's method⁸ does not impose accurately the desired Lagrangian velocity on the immersed boundary ($U_{IB} = 0$), as shown in Fig. 8a. A better improvement is noticed for Pinelli et al.¹⁸ and Cheylan et al.¹⁷, as illustrated respectively in Figs. 8b and 8c, when compared to the multi-DF-IBM variant adopted in Refs. 9–11 as shown in Fig. 8d.

Finally, the results of Ji et al.¹³ displayed in Fig. 8e and the results for any of the cases shown

Article

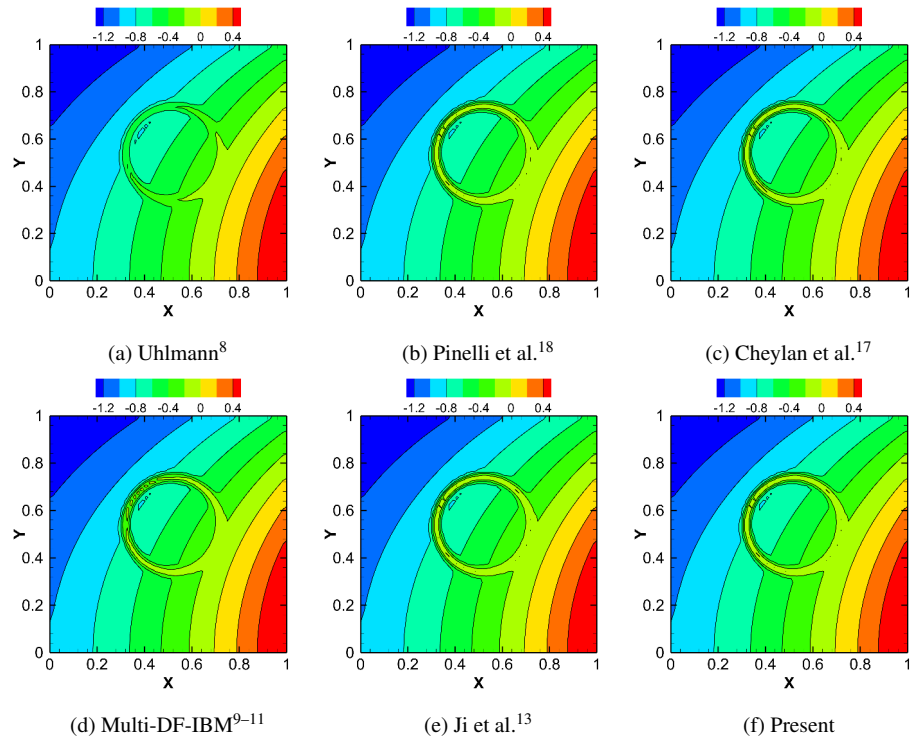


FIG. 8: Velocity contours for different DF-IBM strategies.

in Table VI for the present method revealed in Fig. 8f are identical, and almost indistinguishable from Figs. 8b and 8c from a qualitative point of view, contrary to the quantitative analysis shown in Table VI.

C. No-slip boundary error and flow penetration: Sensitivity analysis

The following investigation differs from the previous one since the latter was used to test the DF-IBM strategies alone for a stationary circular cylinder. Conversely, the forthcoming investigation exploits the proposed DF-IBM inside the PISO algorithm for transient flows to study the no-slip boundary error sensitivity over different parameters and demonstrate the overall improvements made. Thus, this comparative study considers the benchmark problem of the unsteady flow over a stationary circular cylinder.

Article

In this investigation, a circular cylinder of diameter $D = 1 \text{ m}$ located at the center $(0,0)$ of an Eulerian squared domain $\Omega_f = [-2.5, 2.5] \text{ m} \times [-2.5, 2.5] \text{ m}$ is considered, covered by 250×250 Eulerian grid points, with $h = 0.02 \text{ m}$. The cylinder is stationary, i.e., $\mathbf{U}_{IB}$ is set to 0 in all directions, and is formed by 156 Lagrangian markers uniformly distributed over Γ_s , with $\Delta s \approx 0.020138 \text{ m}$. $Re = u_\infty D/\nu$ is set to 100, with the inlet freestream velocity $u_\infty = 1 \text{ m/s}$ and the kinematic viscosity ν equal to $0.01 \text{ m}^2/\text{s}$. For the velocity, a Dirichlet boundary condition ($u_\infty = 1 \text{ m/s}, v_\infty = 0$) 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 the advection velocity, i.e., the component of the velocity normal to the boundary. Slip boundary conditions were set to the lateral sides of the domain. For the pressure, Neumann boundary conditions ($\partial p/\partial \mathbf{n} = 0$) are applied on the inlet and lateral sides of the Eulerian domain, while a zero pressure ($p = 0$) is specified at the outflow boundary, as shown in Fig. 12a. The time-step is set to $\Delta t = 0.004 \text{ s}$ leading to an initial $CFL_{ini} = 0.2$ for all simulations. The time is advanced until the non-dimensional time $\frac{tu_\infty}{D}$ reaches a value of 10. The tolerance for Eq. (35) for two successive DF-IBM iterations was set to 1×10^{-8} and ϕ_1 was selected as the kernel function.

Lagrangian Force Initialization Effect:

Firstly, the Lagrangian force $\mathbf{F}_{IB}^{(m-1)}$ found in Eq. (31) of the current DF-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 schemes. The first is done by static initialization, i.e., 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 through dynamic initialization of the variable, which means that the variable at the current time level is assigned a 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. 9a shows the no-slip boundary error in terms of L_2 norm, as computed in Eq. (42) against the non-dimensional time. Fig. 9b shows the number of iterations to reach the tolerance for two successive DF-IBM iterations against the non-dimensional time. It appears clearly that the dynamic initialization scheme speeds up the calculations to reach the prescribed tolerance, by reducing the number of DF-IBM iterations from 414 for the static initialization case to 108 for the dynamic initialization case at $\frac{tu_\infty}{D} = 10$. In addition, the no-slip error is reduced by one order of magnitude from 3.81×10^{-6} for the static case to 4.74×10^{-7} for the dynamic case. Moreover, an

Article

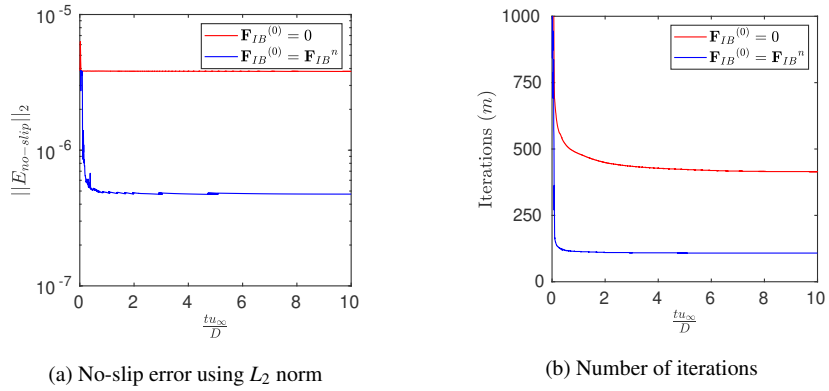


FIG. 9: Effect of the Lagrangian force initialization approach on the no-slip boundary error.

asymptotic behavior for both the number of iterations and the no-slip boundary error against the non-dimensional time can be observed, where a slight change occurred from $\frac{t u_\infty}{D} = 1$, with an error of 3.82×10^{-6} corresponding to 483 iterations, to $\frac{t u_\infty}{D} = 10$, with a recorded error of 3.81×10^{-6} and 414 iterations for the static case.

Similarly for the dynamic case, the error went from 4.87×10^{-7} to 4.74×10^{-7} and the number of iterations decreased from 114 at $\frac{t u_\infty}{D} = 1$ to 108 at $\frac{t u_\infty}{D} = 10$. The dynamic approach will be used in the remaining part of this investigation. It is also worth mentioning that the aerodynamic forces are identical for both initialization schemes.

Lagrangian Weight Method and Lagrangian Weight Correction Effect:

The correlation between the Lagrangian weight method, the Lagrangian weight correction, and the no-slip boundary error is illustrated in Fig. 10. The no-slip boundary error in the L_2 norm following Eq. (42) is not heavily affected by the different methods to compute the Lagrangian weight and the inclusion of the proposed correction, since a minor decrease was observed for the cases using Eqs. (27) and (28) compared to Eq. (26). The number of iterations at $\frac{t u_\infty}{D} = 10$ decreased from 114 iterations in the case of W_n being computed according to Uhlmann⁸ using Eq. (26) to approximately 86 iterations in the case of using Eqs. (27) and (28). This number was further decreased to roughly 70 iterations when the analytical correction was applied. The difference between the results of Eqs. (27) and (28) are barely distinguishable when no correction was used, similar to the case where the correction was applied. This decrease of almost 40% of the

Article

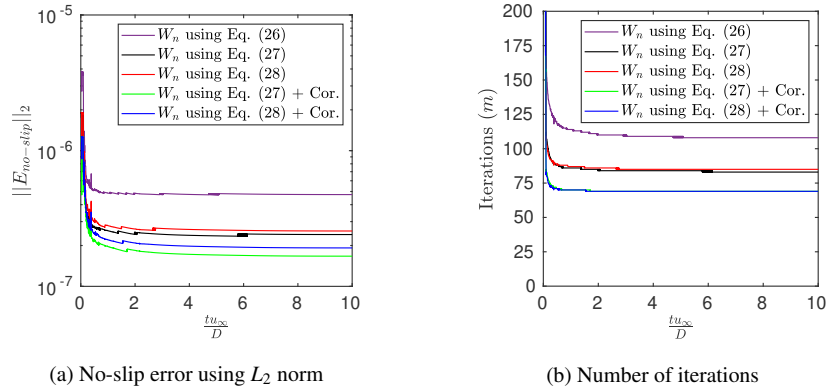


FIG. 10: Effect of the Lagrangian weight method and the Lagrangian weight correction on the no-slip boundary error.

number of iterations per time-step to reach the prescribed tolerance on two successive iterations by employing the proposed correction on the Lagrangian weight Eqs. (27) and (28) compared to Eq. (26) will reflect a large decrease in the CPU time to reach the simulation end time.

Similarly to the previous investigation, an asymptotic behavior was observed for both the no-slip error and the number of iterations, and the aerodynamic forces are also identical for all Lagrangian weight methods, with or without correction.

Time-Step Effect:

A final investigation on the effect of the simulation time-step, i.e., the initial CFL number, on the no-slip boundary error and the number of iterations is presented in Fig. 11. Different values of the time-step were selected, $\Delta t = 0.01, 0.008, 0.006, 0.004$, and 0.002 s, leading to an initial Courant number of $CFL_{ini} = 0.5, 0.4, 0.3, 0.2$, and 0.1 , respectively. The no-slip boundary error and the number of iterations are reduced with decreasing values of the CFL number, in agreement with previous findings in the literature^{10,16}. Unfortunately, manipulating the time-step to reduce the no-slip boundary error is not considered the best approach, since this will drastically increase the CPU time to reach the simulation end time, once Δt is set to a very small value.

Article

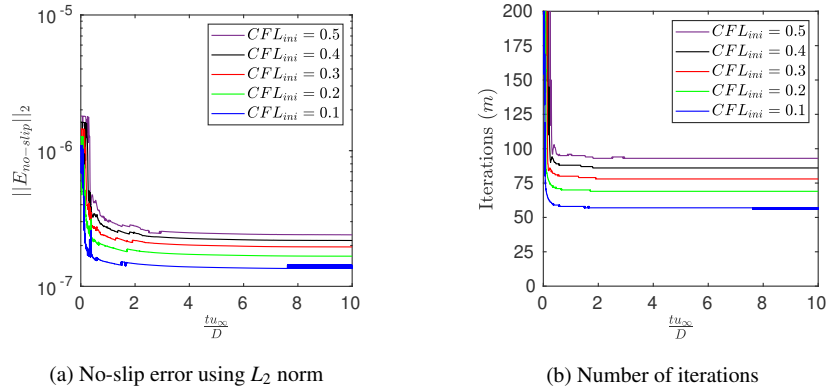


FIG. 11: Effect of the simulation time-step on the no-slip boundary error.

D. Flow over a stationary cylinder at $Re = 40, 100, 300$, and 1000

The current algorithm is validated against well-established experimental and numerical results for the benchmark problem of the steady and unsteady flows over a stationary circular cylinder, over a wide range of Reynolds numbers $Re = u_\infty D / \nu$, with u_∞ the freestream inlet velocity, D the cylinder diameter, and ν the kinematic viscosity. The cylinder diameter and inlet velocity are fixed to 1, while the kinematic viscosity is varied to obtain the simulated Re numbers, i.e., 40, 100, 300, and 1000. Two Eulerian squared domains are used for the current simulations, $\Omega_{f1} = 40D \times 40D = [-15, 25] \times [-20, 20]$ and an enlarged one $\Omega_{f2} = 60D \times 60D = [-20, 40] \times [-30, 30]$, to test the finite lateral domain boundaries confinement effect. The challenge is to reduce the computational time, especially when Re is set equal to 1000. Considering that only the region in the vicinity of the immersed boundary needs to be surrounded by Cartesian uniform cells, for the IBM-related operators. A good strategy is to avoid meshing the whole computational domain with Cartesian uniform cells, and create only a region enclosing the cylinder's immersed boundary with a resolution equal to the Lagrangian markers spacing Δs . This enables the reduction of the mesh size in the far-field regions of the Eulerian domain. Hence, a refined grid for every Re number is adopted in the region $[-5, 5] \times [-5, 5]$, enclosing the cylinder. The Eulerian grid resolution is shown in Table VII.

The same boundary conditions as the previous problem are used here as shown in Fig. 12a. The cylinder center is located at $(0, 0)$ where the numbers of uniformly distributed Lagrangian markers

Article

TABLE VII: Eulerian and Lagrangian grids resolution for the flow over a stationary cylinder at different Re .

Re	h (m)	Δs (m)	$\Delta s/h$	N_{Lag}	Δt (s)
40	0.03	0.030207	1.006920	104	0.006
100	0.03	0.030207	1.006920	104	0.006
300	0.02	0.020138	1.006920	156	0.004
1000	0.01	0.010069	1.006920	312	0.002

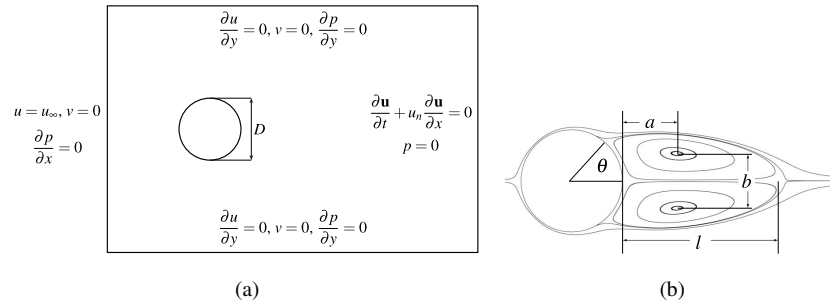


FIG. 12: Computational domain (a) and wake nomenclature (b) for flow over a stationary cylinder.

forming Γ_s for every Re are shown in Table VII with $\Delta s/h \approx 1$ for all conducted simulations. A time-step was selected according to Table VII leading to a maximum Courant number $CFL_{max} \approx 0.911$. The 3-point-width kernel function ϕ_1 was selected for the data mapping between the two grids.

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

$$\text{Drag coefficient: } C_D = \frac{F_D}{0.5\rho u_\infty^2 D},$$

$$\text{Lift coefficient: } C_L = \frac{F_L}{0.5\rho u_\infty^2 D},$$

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

where the fluid density ρ is set to 1 kg/m^3 since we are working with incompressible flows, f_s is the natural vortex shedding frequency expressed in $[1/s]$, and (F_D, F_L) are the respective drag and

Article

lift forces exerted on the cylinder's immersed boundary by the fluid expressed in Newton per unit density [m^4/s^2], keeping in mind that the third dimension is taken equal to 1 since we are working in 2D. They are computed by summing the singular Lagrangian forces located on each Lagrangian marker:

$$\begin{pmatrix} F_D \\ F_L \end{pmatrix} = - \sum_{n=1}^{N_{Lag}} \mathbf{F}_{IB}(\mathbf{X}_n) W_n. \quad (43)$$

The computational time for all simulations with the Eulerian domain Ω_{f1} varied between 11 hours for $Re = 40$ and 21 hours for $Re = 100$ when performed on 12 cores with a mesh size of almost 500k cells. For higher Re , the mesh size becomes significant, therefore 18 cores were used, resulting in a total of 70 hours of CPU time for $Re = 300$ with a grid containing roughly a million cells and 508 hours for $Re = 1000$ having more than 3 million cells.

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, as described in Fig. 12b, with the following dimensions in [m]: 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 θ [$^\circ$] 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. 13. 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, see Table VIII. As can be seen, our findings are consistent with the literature. The drag and lift coefficients are plotted in Fig. 17a, and the steady-state drag value is also compared with values extracted from the literature in Table VIII. It is worth mentioning that by using a larger domain, i.e., by increasing the lateral dimensions of the computational domain to minimize the confinement effect of the lateral boundaries, a reduction of roughly 1.2% was recorded on the drag coefficient value.

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. 14. The vortex shedding leads to time-varying fluctuations of the drag and lift coefficients, and they can be well observed in Fig. 17b. Different force quantities obtained for the current simulations including the mean drag coefficient, the drag and lift fluctuations, and the Strouhal number, are compared in Table IX against different experimental and numerical results reported in the literature for this Re number. These quantities are computed after a periodic state has been achieved by the vortex shedding. A close agreement can be found when compared with previous

Article

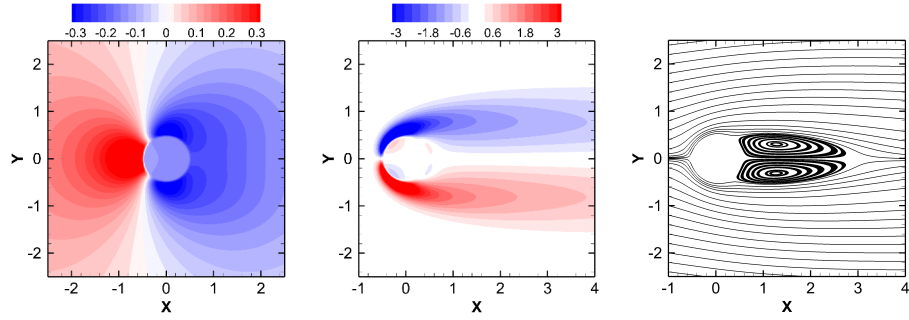


FIG. 13: Instantaneous pressure contours (left), vorticity contours (middle), and streamlines (right) for the flow over a stationary cylinder at $Re = 40$.

TABLE VIII: 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 ^{36a}	—	—	—	—	1.585
Coutanceau & Bouard ^{37a}	2.13	0.76	0.59	53.5	—
Russell & Wang ³⁸	2.29	—	—	53.1	1.600
Linnick & Fasel ³⁹	2.23	0.71	0.59	53.4	1.610
Xu & Wang ⁴⁰	2.21	—	—	53.5	1.660
Taira & Colonius ¹²	2.33	0.75	0.60	54.1	1.550
Wang & Zhang ³⁵	2.36	0.72	0.60	53.8	1.540
Cai et al. ¹⁴	2.36	0.75	0.62	52.1	1.560
Present ($\Omega_{f_1} = 40D \times 40D$)	2.28	0.78	0.61	52.6	1.568
Present ($\Omega_{f_2} = 60D \times 60D$)	2.24	0.78	0.61	52.0	1.550

^a Experimental results.

results. Similar to the steady-state smaller $Re = 40$, a reduction of almost 1.7% is obtained on the mean drag coefficient by enlarging the Eulerian domain lateral dimensions, along with a decrease of 3% for the Strouhal number.

The periodic von Kármán vortex shedding is still present at $Re = 300$, see Fig. 15, along with

Article

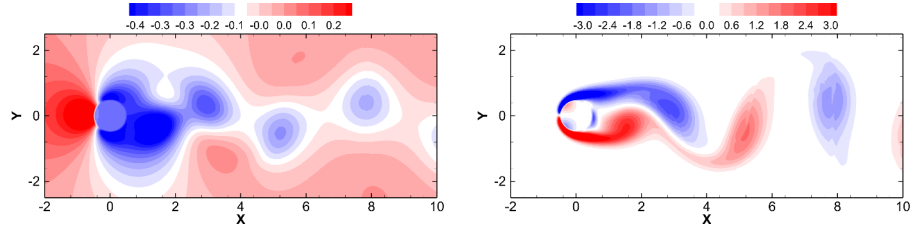


FIG. 14: Instantaneous pressure contours (left) and vorticity (right) contours for the flow over a stationary cylinder at $Re = 100$.

TABLE IX: 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 ^{41a}	—	—	—	0.166
Braza et al. ⁴²	1.364	± 0.015	± 0.250	0.160
Lai & Peskin ⁴³	1.447	—	± 0.329	0.165
Uhlmann ⁸	1.453	± 0.011	± 0.339	0.169
Yang et al. ²⁷	1.393	—	± 0.335	0.165
Ji et al. ¹³	1.402	± 0.010	± 0.349	0.167
Mimeau et al. ⁴⁴	1.400	± 0.010	± 0.320	0.165
Cai et al. ¹⁴	1.366	± 0.010	± 0.342	0.160
Cheyhan et al. ¹⁷	1.408	—	± 0.330	0.166
Present ($\Omega_{f_1} = 40D \times 40D$)	1.411	± 0.012	± 0.371	0.165
Present ($\Omega_{f_2} = 60D \times 60D$)	1.387	± 0.011	± 0.359	0.160

^a Experimental results.

the pressure contours. At this Re and higher values, the numerical results are expected to deviate from experimental values, due to the intrinsic three-dimensional effects of the flow around the cylinder⁴⁵ leading to inaccurate prediction of the forces coefficients for two-dimensional numerical simulations. The time evolution of the drag and lift coefficients caused by the vortex shedding are plotted in Fig. 17c. Table X compares the corresponding mean drag coefficient, the rms of the

Article

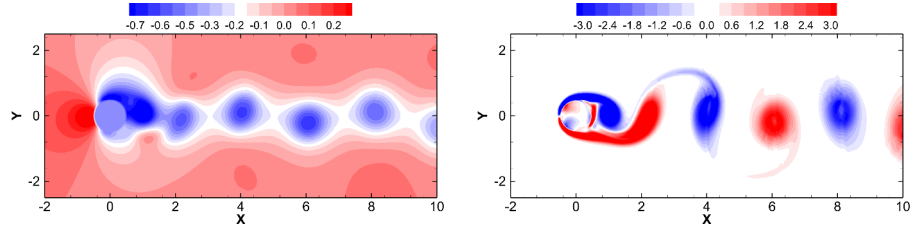


FIG. 15: Instantaneous pressure contours (left) and vorticity contours (right) for the flow over a stationary cylinder at $Re = 300$.

TABLE X: Comparison of the mean drag coefficient, the root mean square of the lift fluctuations, and the Strouhal number for the flow over a stationary cylinder at $Re = 300$.

References	$\overline{C_D}$	$(C_L)_{rms}$	Str
Henderson ⁴⁶	1.370	—	—
Mittal & Balachandar ⁴⁵	1.381	0.652	0.213
Persillon & Braza ⁴⁷	1.405	0.526	0.209
Ye et al. ⁴⁸	1.380	—	0.210
Mittal et al. ⁴⁹	1.360	—	0.210
Apte et al. ⁵⁰	1.410	—	0.212
Present ($\Omega_{f_1} = 40D \times 40D$)	1.501	0.691	0.210
Present ($\Omega_{f_2} = 60D \times 60D$)	1.484	0.680	0.210

lift fluctuations, and the Strouhal number for the current solver against various two-dimensional simulation results extracted from the literature. The mean value of the drag coefficient for all six references is equal to 1.384, leading to a discrepancy of about 8.45% and 7.22% with our computed result for Ω_{f_1} and Ω_{f_2} , respectively. The mean Strouhal number for all mentioned references is approximately equal to 0.211, resulting in a difference of only 0.47% with our calculated Strouhal number, which remained intact for different dimensions of the domain.

At $Re = 1000$, the flow regime becomes dominant by convection effects, leading to a decrease of the boundary layer thickness, expressed as in Ref. 44, by $\delta \approx D/\sqrt{Re} \approx 0.0316 m$. Hence, to marginally capture the thin boundary layer, 3 Eulerian grid cells are considered inside the boundary layer for this case, leading to an Eulerian grid resolution of $h = 0.01 m$ in Table VII, similarly

Article

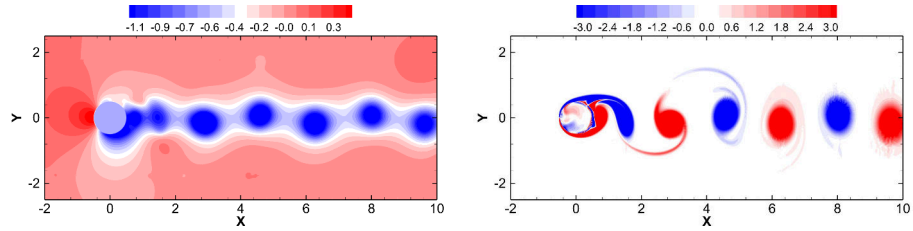


FIG. 16: Instantaneous pressure contours (left) and vorticity contours (right) for the flow over a stationary cylinder at $Re = 1000$.

TABLE XI: Comparison of the mean drag coefficient, the drag and lift fluctuations amplitude, and the Strouhal number for the flow over a stationary cylinder at $Re = 1000$.

References	$\overline{C_D}$	C'_D	C'_L	Str
Henderson ⁴⁶	1.510	—	—	—
Mittal & Kumar ⁵¹	1.480	± 0.210	± 1.360	0.250
Mittal et al. ⁴⁹	1.450	—	—	—
Apte et al. ⁵⁰	1.500	—	—	0.238
Mimeau et al. ⁴⁴	1.510	± 0.230	± 1.540	0.245
Cai et al. ¹⁴	1.550	± 0.220	± 1.460	0.240
Present ($\Omega_{f_l} = 40D \times 40D$)	1.579	± 0.227	± 1.353	0.255

to previous research^{49,50}. Despite the boundary layer being partially resolved, the flow characteristics were well depicted, see Fig. 16 for the instantaneous pressure and vorticity fields. The time evolution of the drag and lift coefficients are plotted in Fig. 17d. The two-dimensional mean drag coefficient, the drag and lift fluctuations, and the Strouhal number are compared against the numerical results reported in the literature in Table XI. A difference of 5.26% was recorded for the drag coefficient when compared with the average value of all literature values, whereas a 4.84% deviation was noted for the Strouhal number.

Overall, the present solver has demonstrated strong agreement with experimental results and other numerical simulations, further validating and confirming its accuracy across a relatively broad range of Reynolds numbers.

Article

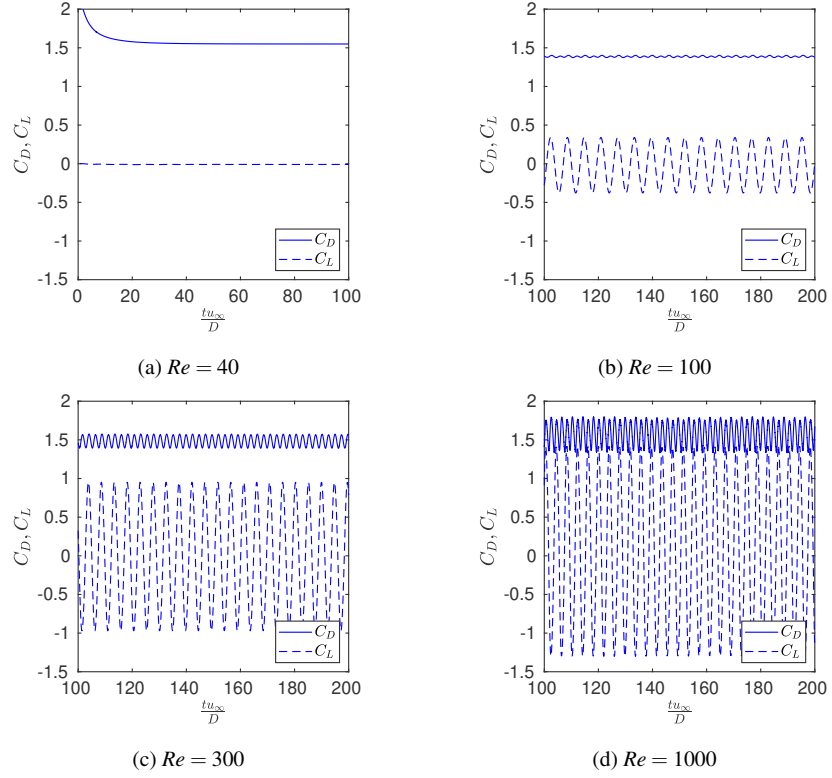


FIG. 17: Temporal evolution of the drag C_D and lift C_L coefficients for the flow over a stationary cylinder at different Re .

E. Flow over an in-line oscillating cylinder in a quiescent fluid at $Re = 100$

The flow produced by the periodic streamwise oscillation of a circular cylinder is now simulated using the present method. In this test case, the fluid is considered stagnant, and the in-line translational motion of the cylinder is prescribed by the following equation:

$$x(t) = -A \sin(2\pi f t), \quad (44)$$

where x [m] is the location of the cylinder's center in the longitudinal direction, A [m] is the amplitude of the oscillation, and f [1/s] is the characteristic frequency.

This flow is characterized by the Reynolds number defined as $Re = U_{max} D / \nu$, and the Keulegan-

Article

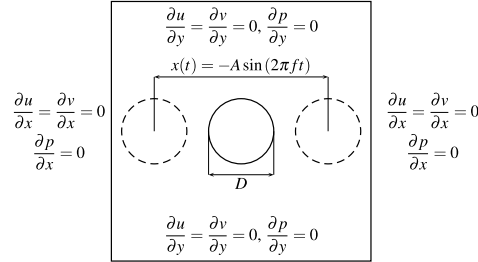


FIG. 18: Computational domain and boundary conditions for flow over an in-line oscillating cylinder.

Carpenter number $KC = U_{max}/fD$, where $U_{max} = 2\pi Af$ is the maximum velocity of the cylinder during oscillation, D is the cylinder diameter, and ν is the kinematic viscosity. The computations are therefore performed at $Re = 100$ and $KC = 5$ for this test case, with $D = 1$ m, $\nu = 0.01$ m²/s, $U_{max} = 1$ m/s, $A = 2.5/\pi$ m, and $f = 0.2$ 1/s. Experimental and numerical results obtained by Dütsch et al.⁵² are used for comparison.

An Eulerian squared domain is used, with dimensions $\Omega_f = 40D \times 30D = [-20, 20] \times [-15, 15]$. To reduce the computational time similarly to the previous problem, the region $[-5, 5] \times [-5, 5]$ near the cylinder boundary that covers its minimum and maximum displacement was created with an Eulerian grid resolution of $h = 0.025$ m. Neumann boundary conditions are applied to both pressure ($\partial p / \partial \mathbf{n} = 0$) and velocity ($\partial \mathbf{u} / \partial \mathbf{n} = 0$) for all domain boundaries, as shown in Fig. 18. The cylinder is located initially at the center of Ω_f , i.e., $(0, 0)$, and Γ_s is formed by 124 Lagrangian markers having the arc length between two consecutive markers $\Delta s \approx 0.025335$ m. The desired Lagrangian velocity of the cylinder boundary $\mathbf{U}_{IB}$ is enforced at each time-step by taking the derivative of Eq. (44) with respect to time. A constant time-step of $\Delta t = 0.005D/U_{max}$ is used, similarly to Ref. 53, and the kernel function ϕ_2 is selected for the interpolation and spreading operators.

Fig. 19 demonstrates the pressure and vorticity contours at four different phase angles, $\varphi = 0^\circ, 96^\circ, 192^\circ$, and 288° (having $\varphi = t/T \times 360^\circ$). Good agreement is observed qualitatively with the numerical results obtained by Dütsch et al.⁵², and a similar flow structure for the two counter-rotating vortices of equal strength is illustrated. Another quantitative comparison is carried out with the experimental results extracted by Dütsch et al.⁵² for the velocity profiles for both u and v components, at three different phase angles $\varphi = 180^\circ, 210^\circ$, and 330° , for four different

Article

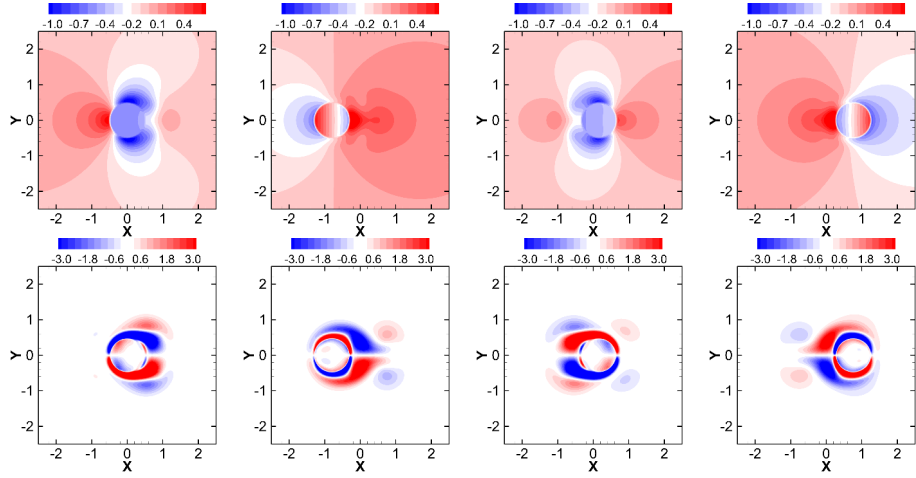


FIG. 19: Instantaneous pressure contours (top) and vorticity contours (bottom) for different phase angle $\varphi = 0^\circ, 96^\circ, 192^\circ$, and 288° (from left to right) for the flow over an in-line oscillating cylinder in a quiescent fluid at $Re = 100$ and $KC = 5$.

streamwise cross-sections $x = -0.6D, 0D, 0.6D$, and $1.2D$. Fig. 20 shows the results obtained by the current approach in comparison with the experimental results. It can be concluded that the computed results for the velocity profile are consistent and align well with the experimental data. This example demonstrates that the present solver can properly capture the dynamics of the flow field as well as the velocity distributions for moving immersed boundaries.

F. Flow over a transversely oscillating cylinder in a freestream at $Re = 185$

Another cylinder oscillating case is presented, where the cylinder is now oscillating in the crosswise direction of a freestream going from the left to the right boundaries of the Eulerian domain. The crosswise translational motion of the cylinder is prescribed by the following equation:

$$y(t) = A \sin(2\pi ft), \quad (45)$$

with y [m] the lateral position of the cylinder centroid, A [m] the amplitude of the oscillation, and f [1/s] the characteristic frequency of the oscillation.

The flow's Reynolds number is set to $Re = u_\infty D / \nu = 185$, with $u_\infty = 1$ m/s the uniform freestream velocity, $D = 1$ m the cylinder diameter, and $\nu = 5.4054 \times 10^{-3}$ m²/s the fluid kine-

Article

This is the author's peer reviewed, accepted manuscript. However, the online version of record will be different from this version once it has been copyedited and typeset.

PLEASE CITE THIS ARTICLE AS DOI: 10.1063/5.0231218

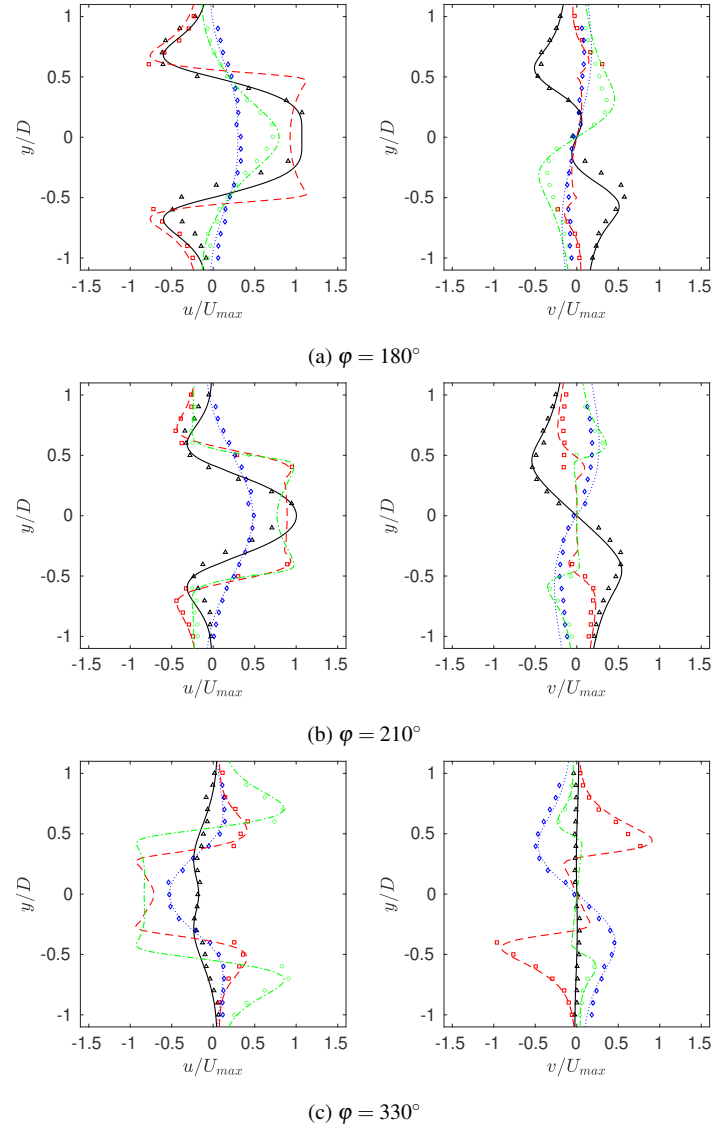


FIG. 20: Non-dimensional velocity profiles for the u (left) and v (right) components at 3 different phase angles ϕ and at 4 different streamwise cross-sections. The current results are marked with lines while the experimental results of Dütsch et al.⁵² are marked with symbols. At $x = -0.6D$ (— and $\triangle$), $x = 0D$ (- - - and $\square$), $x = 0.6D$ (- · - · and $\circ$), and $x = 1.2D$ (· · · · and $\diamond$).

Article

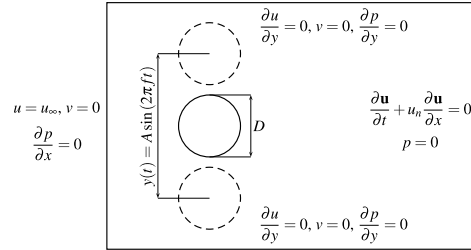


FIG. 21: Computational domain and boundary conditions for flow over a transversely oscillating cylinder.

matic viscosity. The amplitude is fixed to $A = 0.2D$, and the frequency is varied between $0.8 \leq f/f_s \leq 1.2$, with $f_s = 0.195$ 1/s the natural vortex shedding frequency for the stationary cylinder case at $Re = 185$, taken from Ref. 54. Those parameters are considered to compare the results with those extracted from the literature^{55,56}.

The dimensions of the Eulerian domain are defined as $\Omega_f = 60D \times 60D = [-20, 40] \times [-30, 30]$, with a grid resolution $h = 0.02$ m in the refined region $[-5, 5] \times [-5, 5]$, that covers the displacement of the moving cylinder boundary. The boundary conditions are similar to the flow over a stationary cylinder case, see Fig. 21. The cylinder is located at $t = 0$ at the origin of Ω_f , i.e., $(0, 0)$, and Γ_s is represented by 156 Lagrangian markers, with a Lagrangian spacing Δs computed as in Table VII. A time-step of $\Delta t = 0.004$ s is adopted for all simulations. Similarly to the cylinder in-line oscillation problem, the transient desired Lagrangian velocity $\mathbf{U}_{IB}$ is enforced at each time level by taking the derivative of Eq. (45), and the 5-point-width ϕ_2 kernel function was used to reduce spurious force oscillations.

Fig. 22 illustrates the instantaneous vorticity contours when the cylinder is located at the highest positive position. The overall vorticity pattern obtained with the present method when the frequency f/f_s varies, is in agreement with the observations reported in Refs. 55 and 56. The vortex shedding is found to be closer to the wake of the cylinder with increasing f/f_s .

Fig. 23 illustrates the instantaneous temporal evolution of the drag and lift coefficients for all values of the frequencies f/f_s . The challenge was to compute correctly the drag and lift forces exerted on the oscillating cylinder. Due to the nature of the immersed boundary method, a flow is developed inside the immersed object since the interior is treated as part of the Eulerian domain. This internal fluid does not affect the flow outside the immersed object, whereas the forces on the

Article

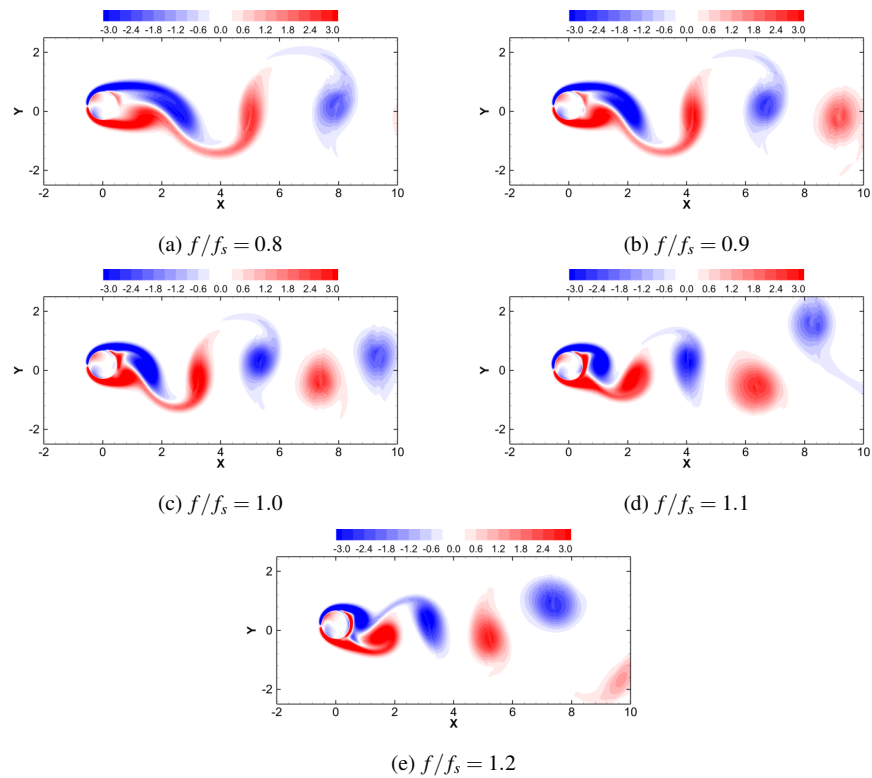


FIG. 22: Instantaneous vorticity contours at different frequencies f/f_s for the flow over a transversely oscillating cylinder in a freestream at $Re = 185$.

immersed boundary are influenced by this internal flow development, i.e., by the internal mass. Once the internal fluid is moved, the internal mass effect should be added to the forces equations to compensate for the accelerating fluid inside the immersed boundary⁵⁷. The patterns of this wide spectrum of frequencies are similar to the one reported in Ref. 55, where a body-conformal method was used, as well as the results obtained for a body non-conforming method in Ref. 56.

A quantitative analysis is also carried out for the mean drag coefficient and the rms values of the drag and lift fluctuations. The findings are presented in Fig. 24. The variation of the mean drag coefficient with increasing f/f_s is in fair agreement with the literature data of Refs. 55 and 56. However, results are slightly higher here, with an average difference of 9.8% when compared with Ref. 55 and 3.8% when compared with Ref. 56 for all frequency ratios combined. Similarly,

Article

This is the author's peer reviewed, accepted manuscript. However, the online version of record will be different from this version once it has been copyedited and typeset.

PLEASE CITE THIS ARTICLE AS DOI: 10.1063/5.0231218

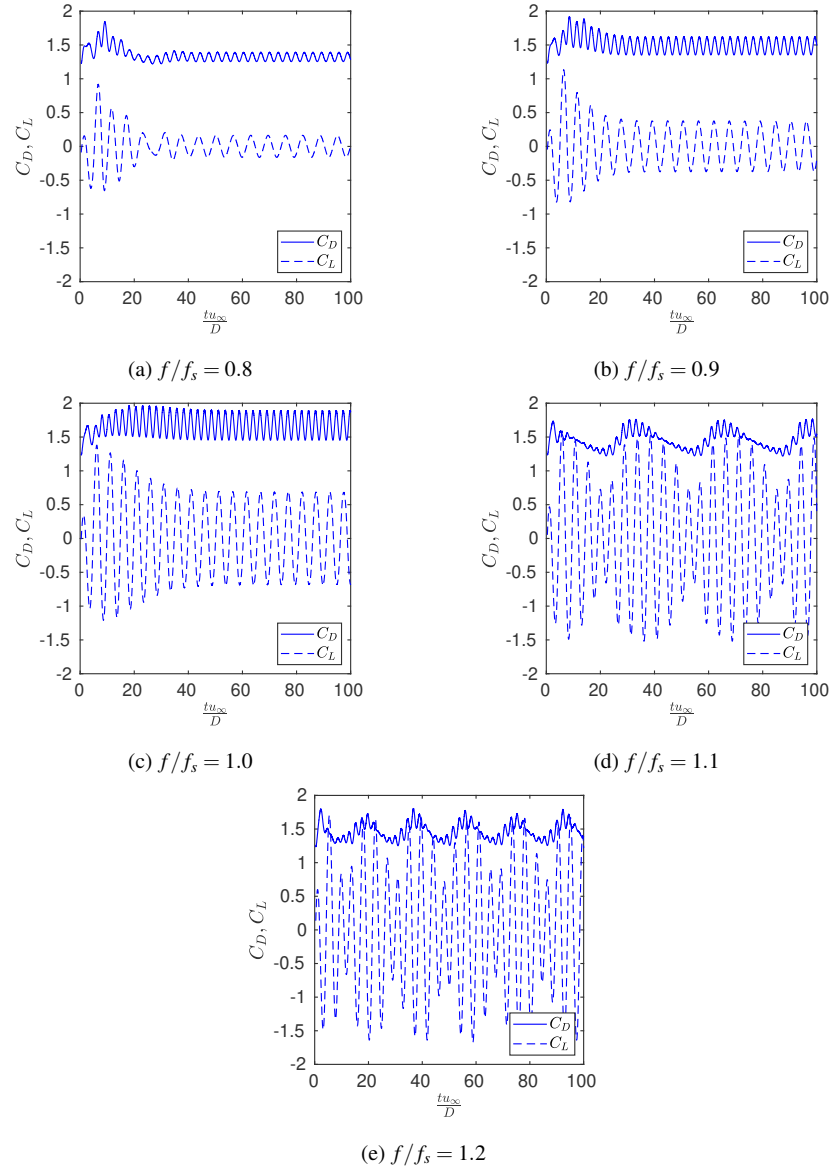


FIG. 23: Temporal evolution of the drag C_D and lift C_L coefficients at different frequencies f/f_s for the flow over a transversely oscillating cylinder in a freestream at $Re = 185$.

Article

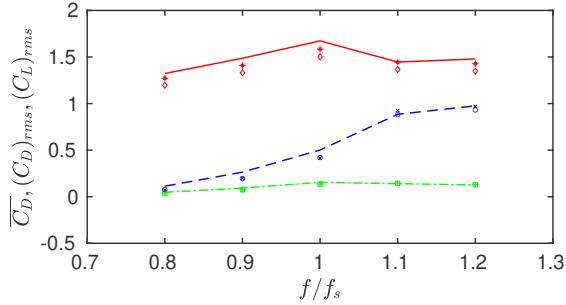


FIG. 24: Comparison of the force coefficients at different frequencies f/f_s for the flow over a transversely oscillating cylinder in a freestream at $Re = 185$. The present results are marked with lines, the results corresponding to Ref. 55 are marked with ($\diamond$; $\square$; $\circ$), and the results corresponding to Ref. 56 are marked with ($*$; $+$; $\times$). $\overline{C_D}$ ($—$, $\diamond$, and $*$); $(C_D)_{rms}$ ($- - -$, $\square$, and $+$); $(C_L)_{rms}$ ($- \cdot - \cdot$, $\circ$, and $\times$).

excellent agreement was found for the rms values of the lift fluctuations for higher frequencies between $1.1 \leq f/f_s \leq 1.2$. For lower frequencies, the computed values were also slightly higher when compared with results in Refs. 55 and 56. The rms values of the drag fluctuations also agree well with the results reported in the literature.

G. Flow over a hovering elliptical wing at $Re = 157$

Finally, the flow produced by a non-circular object in a fluid at rest is simulated, where the immersed boundary is described by an elliptical wing that mimics the shape of an insect's wing, and the hovering of the wing is subject to both translational and rotational prescribed motions governed by the following equations:

$$x(t) = \frac{A_0}{2} \left[\cos\left(\frac{2\pi t}{T}\right) + 1 \right] \cos(\beta), \quad (46)$$

$$y(t) = \frac{A_0}{2} \left[\cos\left(\frac{2\pi t}{T}\right) + 1 \right] \sin(\beta), \quad (47)$$

$$\theta(t) = \frac{\theta_0}{2} \left[1 - \sin\left(\frac{2\pi t}{T} + \varphi_0\right) \right], \quad (48)$$

where (x, y) [m] are the time-dependent ellipse centroid due to motion and θ [rad] is the time-dependent angle of attack of the wing. A_0 and θ_0 designate the translational and rotational amplitudes respectively, β is the stroke angle, φ_0 is the phase difference between the translational

Article

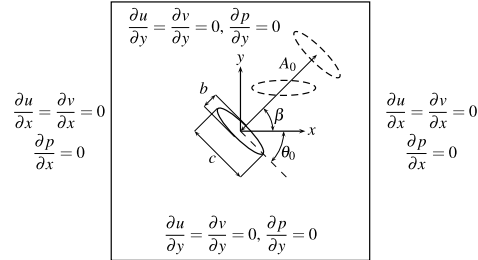


FIG. 25: Computational domain and boundary conditions for flow over a hovering elliptical wing.

and rotational movements, and T is the hovering period. The configuration of the moving wing is described in Fig. 25, with c the major axis (or chord length) and b the minor axis of the ellipse. The flow is characterized by the maximum velocity of translational motion along the hovering path computed as $U_{max} = \pi A_0 / T$. The hovering period is defined by $T = \pi A_0 / c$ and the Reynolds number is $Re = U_{max} c / \nu$. The simulation is performed at $Re = 157$, with $c = 1\text{ m}$, $c/b = 4$, $A_0 = 2.5\text{ m}$, $\theta_0 = \pi/2$, $\beta = \pi/3$, $\varphi_0 = 0$, $U_{max} = 1\text{ m/s}$, and $T = 2.5\pi\text{ s}$, to compare with previous numerical studies from the literature^{14,53,58}.

A smaller squared Eulerian domain is used compared to the recommended one in the works of Refs. 14 and 53, such that $\Omega_f = 20c \times 20c = [-10, 10] \times [10, 10]$, with an Eulerian grid resolution $h = 0.02\text{ m}$ in the region $[-5, 5] \times [-5, 5]$ around the wing minimum and maximum displacement. The domain boundary conditions are similar to the in-line oscillations of a cylinder problem, see Fig. 25. The ellipse is located at the beginning of the simulation at $(0, 0)$ and Γ_s is formed by 106 Lagrangian markers with $\Delta s \approx 0.020232\text{ m}$. The desired Lagrangian velocity of the ellipse boundary $\mathbf{U}_{IB}$ is enforced by summing up the translational and rotational velocities derived from Eqs. (46) to (48). A constant time-step, $\Delta t = 0.004\text{ s}$, was used for this simulation, and ϕ_2 was used as the kernel function to reduce the non-physical spurious force oscillations.

Fig. 26 shows the instantaneous vorticity field produced by the hovering elliptical wing at four different instances of a single flapping period, i.e., $t/T = 0.25, 0.44, 0.74$, and 0.99 . Those correspond to the wing downstroke, where a counter-rotating pair of vortices is generated. These vortices are then shed to generate lift at the half-cycle of the period. Afterward, the wing upstroke is initiated to push a vortex dipole away from the wing. At the end of the period, the vortex dipole is driven further downward of the computational domain. These vorticity contours are very similar to those available in the literature, see Refs. 14, 53, and 58 for instance.

Article

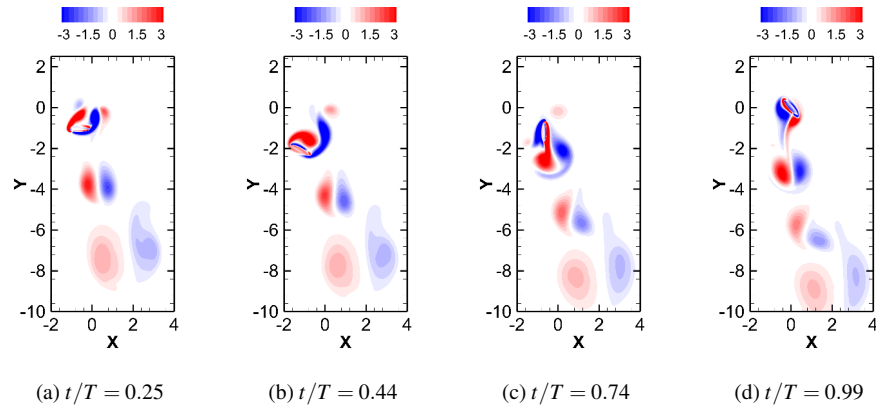


FIG. 26: Instantaneous vorticity contours at four different period instances t/T for the flow over a hovering elliptical wing at $Re = 157$.

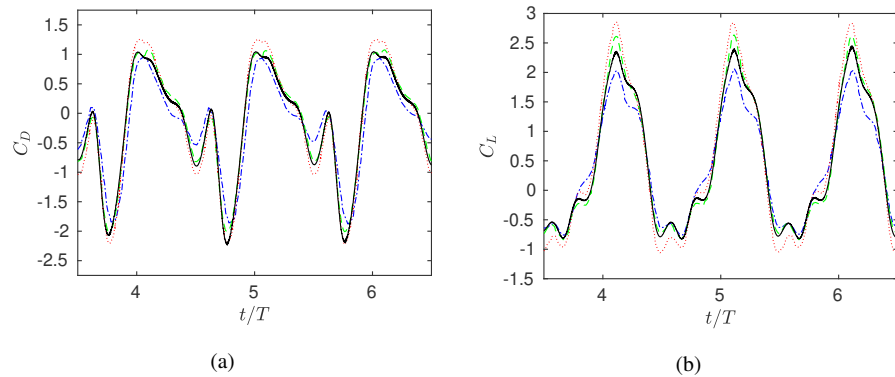


FIG. 27: Temporal evolution of the drag C_D (a) and lift C_L (b) coefficients for the flow over a hovering elliptical wing at $Re = 157$: (—) present results, (.....) moving immersed boundary method (MIBM) of Ref. 14, (---) immersed boundary method of Ref. 53, (-.-.) body-conformal mesh method of Ref. 58.

A quantitative validation of the forces generated by the hovering wing is carried out, by comparing the time history of the instantaneous drag and lift coefficients in Fig. 27, with previous studies computed using a body-conformal mesh method⁵⁸ and body non-conforming immersed boundary methods^{14,53}. The internal mass effect is also considered in this case, similarly to the

Article

previous problem. The data obtained with the present DF-IBM agrees well with other results from the literature, even when performed on a smaller Eulerian domain.

This problem demonstrates that the present method can capture the flow physics even for non-circular objects, while maintaining a good overall accuracy when compared to body-fitted techniques.

VI. CONCLUSIONS

This study has shown improvements to the implicit direct-forcing immersed boundary method (DF-IBM) for incompressible flows simulated past arbitrarily moving rigid structures. The current method leverages the operator-splitting strategy of the original PISO algorithm to address the fluid-solid system's dual constraints problem during the solution process: maintaining the divergence-free condition in the Eulerian fluid domain, and enforcing the no-slip velocity boundary condition in the Lagrangian solid domain. A new pressure Poisson equation (PPE) is derived, which includes the boundary force where the no-slip condition is already satisfied, resulting in a coefficient matrix similar to the standard PPE, i.e., without altering its condition number. This allows the use of fast iterative PPE solvers without any modifications.

The computational cost associated with satisfying the no-slip velocity boundary condition is significantly reduced by integrating Lagrangian weight methods, which enhance the reciprocity of the interpolation-spreading operators, with the implicit formulation of the IBM. Additionally, a Lagrangian boundary force initialization scheme was proposed for the DF-IBM algorithm, to further improve the efficiency of the implicit iterations.

The present method has demonstrated strong capability in handling stationary and moving boundaries in both translational and rotational motions, for both circular and non-circular solid structures. The computed results are consistent and in good agreement with previous experimental and numerical data obtained with conventional body-conforming methods and different IBM variants.

Finally, the proposed DF-IBM variant is characterized by its simplicity of implementation in any existing fluid Navier-Stokes solver. OpenFOAM is recommended as the fluid solver, as it provides a highly scalable platform for future 3D IBM simulations. Future work will consider the coupling between the present DF-IBM and the rigid body dynamics equations of motion to study flow-induced and free motions of rigid structures.

Article

ACKNOWLEDGMENTS

This research is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) [Grant number EP/T518104/1, Project Reference 2676291]. The computations were performed on the platform PILCAM2 from the Université de Technologie de Compiègne, Alliance Sorbonne Université. For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) license to any author accepted manuscript version arising.

DATA AVAILABILITY STATEMENT

The data supporting this work are openly available at <https://doi.org/10.57996/cran.ceres-2582>.

Appendix A: Error norms and order of accuracy

The L_1 , L_2 , and L_∞ norms of the error for any variable x are defined as:

- The L_1 norm:

$$\|E_x\|_1 = \frac{1}{N_x N_y} \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} |x_{i,j} - \hat{x}_{i,j}|, \quad (\text{A1})$$

- The L_2 norm:

$$\|E_x\|_2 = \sqrt{\frac{1}{N_x N_y} \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} |x_{i,j} - \hat{x}_{i,j}|^2}, \quad (\text{A2})$$

- The L_∞ norm:

$$\|E_x\|_\infty = \max_{\substack{1 \leq i \leq N_x \\ 1 \leq j \leq N_y}} |x_{i,j} - \hat{x}_{i,j}|, \quad (\text{A3})$$

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

The order of accuracy of the spatial discretization, i.e., the convergence rate between numerical solutions obtained with refined grids is:

$$\text{Order} = \log_2 \left(\frac{\|E_x\|^{(h)}}{\|E_x\|^{(h/2)}} \right), \quad (\text{A4})$$

Article

where $\|E_x\|^{(h)}$ and $\|E_x\|^{(h/2)}$ are the error norms on a grid size h and a refined one $h/2$. The same formula is used to compute the order of accuracy for the temporal discretization for refined time-steps.

REFERENCES

- ¹T. E. Tezduyar, S. Sathe, R. Keedy, and K. Stein, “Space–time finite element techniques for computation of fluid–structure interactions,” *Computer Methods in Applied Mechanics and Engineering* **195**, 2002–2027 (2006).
- ²T. Nakata and H. Liu, “A fluid–structure interaction model of insect flight with flexible wings,” *Journal of Computational Physics* **231**, 1822–1847 (2012).
- ³S. Roy, A. De, and E. Balaras, *Immersed Boundary Method: Development and Applications*, Computational Methods in Engineering & the Sciences (Springer Nature Singapore, 2020).
- ⁴R. Verzicco, “Immersed Boundary Methods: Historical Perspective and Future Outlook,” *Annual Review of Fluid Mechanics* **55**, 129–155 (2023).
- ⁵C. S. Peskin, “Flow patterns around heart valves: A numerical method,” *Journal of Computational Physics* **10**, 252–271 (1972).
- ⁶M. Souli, A. Ouahsine, and L. Lewin, “ALE formulation for fluid–structure interaction problems,” *Computer Methods in Applied Mechanics and Engineering* **190**, 659–675 (2000).
- ⁷D. Boffi and L. Gastaldi, “Stability and geometric conservation laws for ALE formulations,” *Computer Methods in Applied Mechanics and Engineering* **193**, 4717–4739 (2004).
- ⁸M. Uhlmann, “An immersed boundary method with direct forcing for the simulation of particulate flows,” *Journal of Computational Physics* **209**, 448–476 (2005).
- ⁹K. Luo, Z. Wang, J. Fan, and K. Cen, “Full-scale solutions to particle-laden flows: Multidirect forcing and immersed boundary method,” *Phys. Rev. E* **76**, 066709 (2007).
- ¹⁰T. Kempe and J. Fröhlich, “An improved immersed boundary method with direct forcing for the simulation of particle laden flows,” *Journal of Computational Physics* **231**, 3663–3684 (2012).
- ¹¹W.-P. Breugem, “A second-order accurate immersed boundary method for fully resolved simulations of particle-laden flows,” *Journal of Computational Physics* **231**, 4469–4498 (2012).
- ¹²K. Taira and T. Colonius, “The immersed boundary method: A projection approach,” *Journal of Computational Physics* **225**, 2118–2137 (2007).
- ¹³C. Ji, A. Munjiza, and J. J. R. Williams, “A novel iterative direct-forcing immersed boundary

Article

- method and its finite volume applications,” *Journal of Computational Physics* **231**, 1797–1821 (2012).
- ¹⁴S.-G. Cai, A. Ouahsine, J. Favier, and Y. Hoarau, “Moving immersed boundary method,” *International Journal for Numerical Methods in Fluids* **85**, 288–323 (2017).
- ¹⁵S. Gsell, U. D’Ortona, and J. Favier, “Explicit and viscosity-independent immersed-boundary scheme for the lattice Boltzmann method,” *Phys. Rev. E* **100**, 033306 (2019).
- ¹⁶S. Gsell and J. Favier, “Direct-forcing immersed-boundary method: A simple correction preventing boundary slip error,” *Journal of Computational Physics* **435**, 110265 (2021).
- ¹⁷I. Cheylan, T. Fringand, J. Jacob, and J. Favier, “Analysis of the immersed boundary method for turbulent fluid-structure interaction with Lattice Boltzmann method,” *Journal of Computational Physics* **492**, 112418 (2023).
- ¹⁸A. Pinelli, I. Naqavi, U. Piomelli, and J. Favier, “Immersed-boundary methods for general finite-difference and finite-volume Navier–Stokes solvers,” *Journal of Computational Physics* **229**, 9073–9091 (2010).
- ¹⁹H. G. Weller, G. Tabor, H. Jasak, and C. Fureby, “A tensorial approach to computational continuum mechanics using object-oriented techniques,” *Computer in Physics* **12**, 620–631 (1998).
- ²⁰H. Jasak, H. Weller, and A. Gosman, “High resolution NVD differencing scheme for arbitrarily unstructured meshes,” *International Journal for Numerical Methods in Fluids* **31**, 431–449 (1999).
- ²¹C. W. Gear, “The numerical integration of ordinary differential equations,” *Mathematics of Computation* **21**, 146–156 (1967).
- ²²S. V. Patankar and D. B. Spalding, “A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows,” *International Journal of Heat and Mass Transfer* **15**, 1787–1806 (1972).
- ²³J. P. Van Doormaal and G. D. Raithby, “Enhancements of the SIMPLE method for predicting incompressible fluid flows,” *Numerical Heat Transfer* **7**, 147–163 (1984).
- ²⁴R. I. Issa, “Solution of the implicitly discretised fluid flow equations by operator-splitting,” *Journal of Computational Physics* **62**, 40–65 (1985).
- ²⁵A. M. Roma, C. S. Peskin, and M. J. Berger, “An Adaptive Version of the Immersed Boundary Method,” *Journal of Computational Physics* **153**, 509–534 (1999).
- ²⁶Y. Bao, J. Kaye, and C. S. Peskin, “A Gaussian-like immersed-boundary kernel with three continuous derivatives and improved translational invariance,” *Journal of Computational Physics*

Article

- 316**, 139–144 (2016).
- ²⁷X. Yang, X. Zhang, Z. Li, and G.-W. He, “A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulationsg,” *Journal of Computational Physics* **228**, 7821–7836 (2009).
- ²⁸J. Kim, D. Kim, and H. Choi, “An Immersed-Boundary Finite-Volume Method for Simulations of Flow in Complex Geometries,” *Journal of Computational Physics* **171**, 132–150 (2001).
- ²⁹F. Ilinca and H. J.-F., “Numerical simulation of fluid-solid interaction using an immersed boundary finite element method,” *Computers & Fluids* **59**, 31–43 (2012).
- ³⁰B. E. Griffith and X. Luo, “Hybrid finite difference/finite element immersed boundary method,” *International Journal for Numerical Methods in Biomedical Engineering* **33**, e2888 (2017).
- ³¹J. Favier, A. Revell, and A. Pinelli, “A Lattice Boltzmann–Immersed Boundary method to simulate the fluid interaction with moving and slender flexible objects,” *Journal of Computational Physics* **261**, 145–161 (2014).
- ³²K. Goncharuk, O. Oshri, and Y. Feldman, “The immersed boundary method: A SIMPLE approach,” *Journal of Computational Physics* **487**, 112148 (2023).
- ³³E. Constant, J. Favier, M. Meldi, P. Meliga, and E. Serre, “An immersed boundary method in OpenFOAM: Verification and validation,” *Computers & Fluids* **157**, 55–72 (2017).
- ³⁴M. Askarishahi, “Immersed-boundary/soft-sphere method for particle–particle–fluid interaction in a viscous flow: An OpenFOAM solver,” *Advanced Powder Technology* **34**, 104204 (2023).
- ³⁵S. Wang and X. Zhang, “An immersed boundary method based on discrete stream function formulation for two- and three-dimensional incompressible flows,” *Journal of Computational Physics* **230**, 3479–3499 (2011).
- ³⁶D. J. Tritton, “Experiments on the flow past a circular cylinder at low Reynolds numbers,” *Journal of Fluid Mechanics* **6**, 547–567 (1959).
- ³⁷M. Coutanceau and R. Bouard, “Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation. Part 1. Steady flow,” *Journal of Fluid Mechanics* **79**, 231–256 (1977).
- ³⁸D. Russell and Z. J. Wang, “A Cartesian grid method for modeling multiple moving objects in 2D incompressible viscous flow,” *Journal of Computational Physics* **191**, 177–205 (2003).
- ³⁹M. Linnick and H. Fasel, “A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains,” *Journal of Computational Physics* **204**, 157–192 (2005).

Article

- ⁴⁰S. Xu and Z. J. Wang, "An immersed interface method for simulating the interaction of a fluid with moving boundaries," *Journal of Computational Physics* **216**, 454–493 (2006).
- ⁴¹C. H. K. Williamson, "Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low Reynolds numbers," *Journal of Fluid Mechanics* **206**, 579–627 (1989).
- ⁴²M. Braza, P. Chassaing, and H. H. Minh, "Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder," *Journal of Fluid Mechanics* **165**, 79–130 (1986).
- ⁴³M.-C. Lai and C. S. Peskin, "An Immersed Boundary Method with Formal Second-Order Accuracy and Reduced Numerical Viscosity," *Journal of Computational Physics* **160**, 705–719 (2000).
- ⁴⁴C. Mimeau, F. Gallizio, G.-H. Cottet, and I. Mortazavi, "Vortex penalization method for bluff body flows," *International Journal for Numerical Methods in Fluids* **79**, 55–83 (2015).
- ⁴⁵R. Mittal and S. Balachandar, "On the inclusion of three-dimensional effects in simulations of two-dimensional bluff-body wake flows," *Proceedings of the 1997 ASME Fluids Engineering Division Summer Meeting* (1997).
- ⁴⁶R. D. Henderson, "Details of the drag curve near the onset of vortex shedding," *Physics of Fluids* **7**, 2102–2104 (1995).
- ⁴⁷H. Persillon and M. Braza, "Physical analysis of the transition to turbulence in the wake of a circular cylinder by three-dimensional Navier–Stokes simulation," *Journal of Fluid Mechanics* **365**, 23–88 (1998).
- ⁴⁸T. Ye, R. Mittal, H. S. Udaykumar, and W. Shyy, "An Accurate Cartesian Grid Method for Viscous Incompressible Flows with Complex Immersed Boundaries," *Journal of Computational Physics* **156**, 209–240 (1999).
- ⁴⁹R. Mittal, H. Dong, M. Bozkurtas, F. M. Najjar, A. Vargas, and A. Von Loebbecke, "A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries," *Journal of Computational Physics* **227**, 4825–4852 (2008).
- ⁵⁰S. Apte, M. Martin, and N. Patankar, "A numerical method for fully resolved simulation (FRS) of rigid particle-flow interactions in complex flows," *Journal of Computational Physics* **228**, 2712–2738 (2009).
- ⁵¹S. Mittal and V. Kumar, "Flow-induced vibrations of a light circular cylinder at reynolds numbers 10^3 to 10^4 ," *Journal of Sound and Vibration* **245**, 923–946 (2001).
- ⁵²H. Dütsch, F. Durst, S. Becker, and H. Lienhart, "Low-Reynolds-number flow around an oscillating circular cylinder at low Keulegan–Carpenter numbers," *Journal of Fluid Mechanics* **360**,

This is the author's peer reviewed, accepted manuscript. However, the online version of record will be different from this version once it has been copyedited and typeset.

PLEASE CITE THIS ARTICLE AS DOI: 10.1063/5.0231218

Article

- 249–271 (1998).
- ⁵³J. Yang and F. Stern, “A simple and efficient direct forcing immersed boundary framework for fluid–structure interactions,” *Journal of Computational Physics* **231**, 5029–5061 (2012).
- ⁵⁴X.-Y. Lu and C. Dalton, “Calculation of the timing of vortex formation from an oscillating cylinder,” *Journal of Fluids and Structures* **10**, 527–541 (1996).
- ⁵⁵E. Guilmineau and P. Queutey, “A Numerical Simulation of vortex shedding from an oscillating circular cylinder,” *Journal of Fluids and Structures* **16**, 773–794 (2002).
- ⁵⁶J. Yang and E. Balaras, “An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries,” *Journal of Computational Physics* **215**, 12–40 (2006).
- ⁵⁷Y. Wang, C. Shu, C. J. Teo, and J. Wu, “An immersed boundary-lattice Boltzmann flux solver and its applications to fluid-structure interaction problems,” *Journal of Fluids and Structures* **54**, 440–465 (2015).
- ⁵⁸Z. J. Wang, “Two Dimensional Mechanism for Insect Hovering,” *Phys. Rev. Lett.* **85**, 2216–2219 (2000).



DSPACE

<https://dspace.org/>

An improved implicit direct-forcing immersed boundary method (DF-IBM) around arbitrarily moving rigid structures

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

2024-10-01

American Institute of Physics (AIP)

<https://doi.org/10.1063/5.0231218>

Farah E, Ouahsine A, Verdin PG. (2024) An improved implicit direct-forcing immersed boundary method (DF-IBM) around arbitrarily moving rigid structures. Physics of Fluids, Volume 36, Issue 10, September 2024, Article number 103618