Rigorous benchmarking of an iterative IBM solver by comparison to body-fitted mesh results

Lianxia Li[a,1], Michael Stoellinger[a], Maysam Mousaviraad[b]

[a] *Department of Mechanical Engineering, University of Wyoming, 1000 E. University Ave., Laramie, 82071, WY, USA*
[b] *School of Engineering - Cal Poly Humboldt, 1 Harpst Street, Arcata, 95521, CA, USA*

Abstract

This study addresses the validation challenges of Immersed Boundary Method (IBM) solvers by proposing a consistent benchmarking approach. A temporally second-order Arbitrary Lagrangian-Eulerian (ALE) solver implemented in OpenFOAM is verified against lid-driven cavity flow and flow past static and oscillating cylinders. It then serves as the underlying fluid flow solver for a fully implicit diffusive direct forcing IBM solver that achieves temporal second-order accuracy, verified by the static and oscillating cylinder cases. Leveraging the shared numerical framework, the IBM solver is rigorously validated against ALE using identical settings, parameters, and nearly identical grids across twenty test cases, including stationary cylinder flow (Re = 40, 100, 185), oscillating cylinder in quiescent flow (Re = 100, KC = 5), and transversely oscillating cylinders with varying amplitudes and frequencies (16 cases in total). The IBM solver demonstrates robustness (CFL up to 8) and accuracy over a wide Reynolds number range, with relative drag coefficient errors below 7% for all considered cases. Incorporating body force correction, which reduces slip errors at negligible added computational cost, drastically improves accuracy. Proper Rhie-Chow momentum interpolation, including body forces, is critical in the adopted collocated finite volume method. For unsteady problems, the computational cost is greatly

*Funding: This work is partially funded by the Wind Energy Research Center at the University of Wyoming and the Wyoming INBRE (Grant No. 2P20GM103432)
[1] Corresponding author, lli16@uwyo.edu

reduced without compromising accuracy by using $CFL \approx 1.5 - 2$ with only two outer iterations. The proposed benchmarking framework can be easily used for validation of other IBM approaches and thus potentially help to advance the state of IBM.

Keywords: Immersed Boundary Method, ALE, PISO, Fully implicit,, Force correction, Bechmarking

---

1. Introduction

The Immersed Boundary Method (IBM) in Computational Fluid Dynamics (CFD) has attracted great interest since it was first proposed by Peskin [1]. It does not require a conforming boundary at the interface between fluid and structure domains, which starkly contrasts traditional body-fitted methods such as the Arbitrary Lagrangian-Eulerian (ALE) method. Various IBM methods have been proposed, and they differ mainly in how the body force is derived and imposed and how the interpolation method is used to impose the boundary conditions at the interface. Peskin's original continuous forcing approach [1] adds the body force before discretizing the Naiver-Stokes (NS) equations. In the discrete or direct forcing approach [2, 3], the body force is introduced into the discretized NS equations. The direct forcing approach calculates a body force at the grid point or immediately next to the interface to impose the velocity boundary conditions. Alternatively, the sharp interface method reconstructs the velocity and pressure by interpolation; therefore, the actual body force calculation is unnecessary. More recent developments of IBM can be found in several reviews [4–8].

The verification and validation of IBM poses a significant challenge since rigorous benchmarking is difficult. When using experimental results for benchmarking, the usual issues of experimental uncertainty exist along with the challenges of replicating the physical conditions of the experiment in the simulations. For example, 3D and wind tunnel boundary effects are often significant [9, 10]. Using body-fitted CFD solvers for validation of the IBM methods is more

suitable, but in practice, the benchmark data is often obtained with different numerical methods, different grid resolutions, and usually different domain sizes and boundary conditions. The differences in the numerical settings and simulation details can all cause discrepancies in the results and thus it is impossible to assess the accuracy of the specific IBM approach in isolation or to perform a detailed analysis of the influence of the various aspects of IBM on the overall accuracy.

Validating IBM solvers for moving boundaries is even more limited and mostly restricted to very simple cases for which analytical solutions exist, or to comparisons with other published IBM and experimental results. For example, the slip error resulting from the non-reciprocity of the interpolation and spreading operators in diffusive IBM methods may be considered insignificant, especially for steady-state flows at low Reynolds numbers, but this can lead to reduced accuracy of the IBM solver in other cases [11, 12]. For IBM implemented in a collocated finite volume method (FVM) framework [13–15], the Rhie-Chow interpolation of the body force is another subtle issue that is often overlooked. While the Rhie-Chow interpolation of the body force may have little impact when the body force is small, it could be crucial when it is large. Therefore, to ensure consistency, appropriate Rhie-Chow treatment must be applied to avoid pressure checker-boarding on collocated grids.

Another issue when assessing the accuracy of IBM is temporal discretization. The standard IBM method involves computing the body force using the direct forcing approach as $F = \dfrac{V - U}{\Delta t}$, where V is the velocity of the boundary, U is the intermediate velocity of the fluid flow at the boundary without taking the solid body into account, and Δt is the time step. In explicit IBM, U is computed only once, leading to the body force being fixed during a time step. As the simulation progresses to the next time step and the body force is incorporated into the NS equations, the solid body immediately affects the fluid. Consequently, the intermediate flow velocity U is affected, and the interaction between the fluid and solid must be synchronized for a precise solution. The widely used projection method (fractional step method) based explicit IBM solver [16–18]

typically require much more stringent Courant-Friedrich-Levy (CFL) number limits than dictated by the common convection limit. Thus, very small time steps are needed to achieve accurate results.

Implicit methods allow for larger CFL numbers while maintaining stability. For IBM, this should include an implicit treatment of the body force. Ji et al. [19] presented an iterative IBM that updates the body force within the pressure correction iterations to achieve a more implicit treatment of the body force. Nicolaou et al. [20] split the intermediate velocity equation into two steps. The force is updated iteratively in the second step, but only with the diffusion term, while the convection and pressure terms are frozen. The implicit IBM increases the CFL from 0.5 of the explicit solver to 2.0. Both Ji et al. [19] and Nicolaou et al. [20] used the semi-implicit fractional step method solvers.

The objective of this study is to develop a rigorous and consistent benchmarking strategy for IBM solvers by using an ALE solver within the same numerical framework. To achieve this, we developed an IBM solver on top of a spatially and temporally second-order accurate iterative ALE solver [21] for collocated grids. Maintaining the solver's second-order temporal accuracy for IBM requires outer iterations within a time-step similar to what was done in [19, 20]. Both the IBM and the ALE solvers were implemented in the framework of OpenFOAM (v2106). The only difference between the two solvers is in the treatment of the fluid-solid interface, which is done via body-conforming meshes and a no-slip boundary condition in the ALE solver, using diffusive direct forcing in the IBM solver. Stationary and moving circular cylinders are used as benchmark problems. The domain size, boundary conditions (except for the immersed boundary), and numerical methods used for the ALE and IBM simulations are identical. The grids are also kept as similar as possible and only differ in the immediate region around the fluid-solid interface. Typically, when benchmarking IBM solvers against numerical results, different fluid solvers are employed. For instance, Vreman [22] compared several IBM variants and an overset mesh approach for Stokes flow due to an oscillating sphere, for which an analytical solution exists. Schäfer et al. [23] benchmarked two

IBM solvers from a compact finite-difference code and a pseudo-spectral code against a body-conforming grid solver in the spectral-element code Nek5000. Similarly, [24] conducted a comparison between a high-order spectral-element code (Nek5000), an unstructured finite-volume solver (OpenFOAM), and an in-house IBM Cartesian solver. However, to the best of our knowledge, no prior study has benchmarked an IBM solver against a body-fitted ALE approach implemented within the same numerical framework.

We first verify the temporal second-order accuracy of the IBM solver and then assess its accuracy using the ALE benchmark results. Moreover, we will analyze the relevance of more subtle aspects of IBM, such as the slip error correction and the Rhie-Chow interpolation of the body force. We can also use the benchmark results to optimize the iterative IBM solver settings by determining the required number of outer iterations to achieve a certain accuracy for a given time-step size to reduce the overall computational cost for unsteady simulations.

The paper is organized as follows: Section 2 introduces the key aspects of the iterative ALE solver. The iterative IBM solver, including the body force calculation, force correction technique, and the Rhi-Chow interpolation are discussed in section 3. The ALE solver is verified against static and moving grid cases, including lid-driven cavity flow, flow past a static cylinder, and flow past an oscillating cylinder. The IBM solver is then verified using the same static and oscillating cylinder cases, as detailed in 4. In section 5, the IBM solver is validated using the ALE solver results for a total of twenty cases, including stationary cylinder flow (Re = 40, 100, 185), oscillating cylinder in quiescent flow (Re = 100, KC = 5), and transversely oscillating cylinders with varying amplitudes and frequencies. Various aspects of the IBM solver, including the force correction method, Rhie-Chow treatment for the body force, and computational cost, are discussed in section 6. Section 7 summarizes the findings of this work.

2. ALE solver

We first present the temporally and spatially second-order accurate ALE solver developed by Tuković et al. [21]. This solver forms the baseline fluid flow solver for developing the IBM solver presented in section 3. We omit some of the details presented by Tuković et al. [21] but present the key aspects of the solver algorithm to facilitate the subsequent development of the IBM solver.

2.1. Governing equations

We consider the laminar flow of an incompressible Newtonian fluid with constant density ρ and kinematic viscosity ν inside an arbitrary volume V bounded by a closed moving surface S. The continuity and momentum equations in the integral form are given by

$$\oint \mathbf{n} \cdot \mathbf{u} dS = 0, \tag{1}$$

and

$$\begin{aligned} \frac{d}{dt} \int \mathbf{u}\, dV + \oint \mathbf{n} \cdot (\mathbf{u} - \mathbf{v}_s)\, \mathbf{u}\, dS = \oint \mathbf{n} \cdot (\nu \nabla \mathbf{u})\ dS \\ - \int \nabla p\, dV. \end{aligned} \tag{2}$$

Here, $\mathbf{n}$ is the outward-pointing unit vector normal to S, $\mathbf{u}$ is the fluid velocity, $\mathbf{v}_s$ is the velocity of surface S, and p is the pressure divided by the density. The geometric conservation law (GCL) relates the change of the volume V and the surface velocity $\mathbf{v}_s$

$$\frac{d}{dt} \int dV + \oint \mathbf{n} \cdot \mathbf{v}_s\, dS = 0. \tag{3}$$

2.2. Finite volume discretization

Using the standard finite volume discretization technique for collocated grids, the semi-discretized momentum equation using the second-order backward time discretization scheme is given by

$$\begin{aligned} &\frac{3\,(\mathbf{u}V)_P^{n+1} - 4\,(\mathbf{u}V)_P^{n} + (\mathbf{u}V)_P^{n-1}}{2\Delta t} + \sum_f \left[\left(\dot{V} - \dot{V}_s \right) \mathbf{u} \right]_f^{n+1} \\ &= \sum_f \left[\nu \mathbf{n} \cdot \nabla \mathbf{u}\, S \right]_f^{n+1} - \left(\nabla p\, V \right)_P^{n+1}, \end{aligned} \tag{4}$$

where the subscripts P and f represent cell center and face center values, respectively. The superscripts $(n+1)$, (n), and $(n-1)$ represent values evaluated at the new time instance t^{n+1} and two previous time instance t^n and t^{n-1}, respectively. The volume flow rate due to the fluid flow through the face

$$\dot{V}_f^{n+1} = (\mathbf{n}_f \cdot \mathbf{u}_f \, S_f)^{n+1}, \tag{5}$$

where $\mathbf{u}_f$ is the face center velocity, $\mathbf{n}_f$ is the unit normal vector, and S_f is the face area. The flow rate across the surfaces of the control volume must satisfy the discretized continuity equation (10). Meanwhile, the volume flux due to grid motion $\dot{V}_s^{n+1}$ must satisfy the discretized GCL [25].

To achieve second-order time accuracy, all the flux and source terms must be evaluated at the new time level $(n+1)$. For the non-linear convection term, this is achieved through fixed-point outer iterations [21, 26]

$$\left[\left(\dot{V} - \dot{V}_s\right)\mathbf{u}\right]_f^{n+1} \approx \left(\dot{V}_f^{n+1,i-1} - \dot{V}_{s,f}^{n+1}\right)\mathbf{u}_f^{n+1,i}, \tag{6}$$

where the superscript (i) denotes the outer iterations of solving the momentum equation, and we have assumed that the volume flux due to grid motion $\dot{V}_{s,f}^{n+1}$ is known (i.e., for prescribed solid body motion). It should be noted that Tuković et al. [21] focused on a non-iterative approach that uses an explicit temporal extrapolation to estimate the volume flux at the new time level $\dot{V}_f^{n+1}$. Here, we use the iterative version as shown in Eq. (6).

After choosing the appropriate approximations for the face center velocity $\mathbf{u}_f$ (e.g., linear interpolation from cell values or linear upwind interpolation) and for the face-normal derivative $\mathbf{n}_f \cdot (\nabla \mathbf{u})_f$, the discretized momentum equation can be written in the form of a linear algebraic equation for cell P [21]

$$a_P \mathbf{u}_P^{n+1} + \sum_N a_N \mathbf{u}_N^{n+1} = \mathbf{r}_P - (\nabla p)_P^{n+1}, \tag{7}$$

where a_P is the diagonal coefficient, a_N is the neighbor coefficient and $\mathbf{r}_P$ is the source term.

Following [21], we adopt the Pressure-Implicit Splitting of Operators (PISO) method for solving the coupled system of continuity and momentum equations

in each outer iteration. The momentum equation (7) is solved first using an estimate for the pressure field leading to a velocity field that does not satisfy continuity. A discretized pressure equation is then derived using the discretized momentum and continuity equations including the Rhie-Chow (RC) momentum interpolation method [27] as briefly outlined below. To simplify the presentation, we will first consider a static mesh. In OpenFOAM , the $\mathbf{H}$ operator is introduced which is defined as

$$\mathbf{H}(\mathbf{u}_P) = -\sum a_N \mathbf{u}_N + \mathbf{r}_P = a_P \mathbf{u}_P + (\nabla p)_P \,. \tag{8}$$

After solving the momentum predictor equation (7) with an estimated pressure field, we can now explicitly express the cell center velocity using the $\mathbf{H}$ operator

$$\mathbf{u}_P = \frac{\mathbf{H}(\mathbf{u}_P)}{a_P} - \frac{1}{a_P} (\nabla p)_P \,. \tag{9}$$

The discretized continuity equation for cell P reads

$$\sum_f \mathbf{n}_f \cdot \mathbf{u}_f^{n+1} S_f = \sum_f \dot{V}_f^{n+1} = 0. \tag{10}$$

In the Rhie-Chow momentum interpolation method [21, 27, 28], the cell-face velocity $\mathbf{u}_f$ needed in Eq. (10) is obtained from a "face equivalent" expression of the momentum equation (9) given by

$$\mathbf{u}_f = \left[\frac{\mathbf{H}(\mathbf{u}_P)}{a_P}\right]_f - \left(\frac{1}{a_P}\right)_f (\nabla p)_f. \tag{11}$$

where the coefficients $\left[\frac{\mathbf{H}(\mathbf{u}_P)}{a_P}\right]_f$ and $\left(\frac{1}{a_P}\right)_f$ are obtained by linear interpolation of the corresponding cell-center values as used in Eq. (9). The term $(\nabla p)_f$ is discretized with a "small" stencil using the two cell center values straddling face f. Substituting the expression (11) for the face center velocity into the discrete continuity equation (10) we can derive the discrete pressure equation

$$\sum_f \left(\frac{1}{a_P}\right)_f \mathbf{n}_f \cdot (\nabla p)_f^{n+1} S_f = \sum_f \mathbf{n}_f \cdot \left(\frac{\mathbf{H}(\mathbf{u}_P)}{a_P}\right)_f S_f. \tag{12}$$

For severely non-orthogonal grids, the expression $\mathbf{n}_f \cdot (\nabla p)_f^{n+1}$ is discretized with a deferred-correction approach where the orthogonal part is treated implicitly,

and the non-orthogonal contribution explicitly [29, 30]. In this case, the pressure equation is solved iteratively using a prescribed number of non-orthogonal corrections.

After solving the pressure equation, we can calculate the cell-face volume flow rate that satisfies the continuity equation using

$$\dot{V}_f^{n+1} = \mathbf{n} \cdot \left[\frac{\mathbf{H}(\mathbf{u}_P)}{a_P}\right]_f S_f - \left(\frac{1}{a_P}\right)_f (\mathbf{n} \cdot \nabla p)_f^{n+1} S_f, \tag{13}$$

and update the cell-center velocity using

$$\mathbf{u}_P^{n+1} = \frac{\mathbf{H}\left(\mathbf{u}\right)_P}{a_P} - \frac{1}{a_P}(\nabla p)_P^{n+1}. \tag{14}$$

To establish the connection between the OpenFOAM RC approach to the more commonly adopted explicit RC interpolation

$$\mathbf{u}_f = \overline{\mathbf{u}}_f - \left(\frac{1}{a_P}\right)_f \left[(\nabla p)_f - \overline{(\nabla p)_P}\right], \tag{15}$$

we use Eq. (8) and linearly interpolate it to the face center

$$\left[\frac{\mathbf{H}(\mathbf{u}_P)}{a_P}\right]_f = \overline{\mathbf{u}}_f + \overline{\left(\frac{(\nabla p)_P}{a_P}\right)} \approx \overline{\mathbf{u}}_f + \left(\frac{1}{a_P}\right)_f \overline{(\nabla p)_P}, \tag{16}$$

where the overbar denotes linear interpolation from cell values to face values.

Regrouping terms in (15) and using the above relation, we see that

$$\begin{aligned} \mathbf{u}_f &= \left[\bar{\mathbf{u}}_f + \left(\frac{1}{a_P}\right)_f \overline{(\nabla p)_P}\right] - \left(\frac{1}{a_P}\right)_f (\nabla p)_f \\ &= \left(\frac{\mathbf{H}(\mathbf{u}_P)}{a_P}\right)_f - \left(\frac{1}{a_P}\right)_f (\nabla p)_f, \end{aligned} \tag{17}$$

which is identical to the expression for the face center velocity used in OpenFOAM as given in Eq. (11).

For unsteady flows, Pascau [28] has shown that special care must be taken so that the RC interpolation satisfies the condition of unconditional time-step independence at a steady state. A temporally consistent RC interpolation method for the backward scheme in an ALE solver was presented by Tuković et al. [21]. We closely follow their approach and only present some key expressions here.

The basic idea is to remove the time-step dependence from the $\mathbf{H}$ operator and the diagonal coefficient a_P by introducing

$$a_P = a_P^* + \frac{3}{2\Delta t} \tag{18}$$

$$\mathbf{r}_P = \mathbf{r}_P^* + \frac{4\mathbf{u}_P^n(\frac{V_P^n}{V_P^{n+1}}) - 4\mathbf{u}_P^{n-1}(\frac{V_P^{n-1}}{V_P^{n+1}})}{2\Delta t} \tag{19}$$

$$\mathbf{H} = \mathbf{H}^* + (\mathbf{r}_P - \mathbf{r}_P^*) \tag{20}$$

where the superscript * represents the contribution from convection and diffusion terms independent of the time step and V_P^n denotes the volume of cell P at time t^n. The RC interpolated cell-face velocity $\mathbf{u}_f^{n+1}$ is now given by

$$\begin{aligned}\mathbf{u}_f^{n+1} = &\frac{1}{\tilde{a}_f}\left(\frac{\mathbf{H}^*}{a^*}\right)_f - \frac{1}{\tilde{a}_f}\left(\frac{1}{a^*}\right)_f (\nabla p)_f^{n+1} \\ &+ \frac{1}{\tilde{a}_f}\left(\frac{1}{a^*}\right)_f \frac{4\mathbf{u}_f^n\left(\frac{V^n}{V^{n+1}}\right)_f - \mathbf{u}_f^{n-1}\left(\frac{V^{n-1}}{V^{n+1}}\right)_f}{2\Delta t},\end{aligned} \tag{21}$$

with the coefficient $\tilde{a}_f$ defined by

$$\tilde{a}_f = 1 + \left(\frac{1}{a^*}\right)_f \frac{3}{2\Delta t}. \tag{22}$$

We can write Eq. (21) more compact as

$$\mathbf{u}_f^{n+1} = \left(\frac{\mathbf{H}}{a}\right)_f - \left(\frac{1}{a}\right)_f (\nabla p)_f^{n+1}, \tag{23}$$

where

$$\left(\frac{1}{a}\right)_f = \frac{1}{\tilde{a}_f}\left(\frac{1}{a^*}\right)_f, \tag{24}$$

and

$$\left(\frac{\mathbf{H}}{a}\right)_f = \frac{1}{\tilde{a}_f}\left(\frac{\mathbf{H}^*}{a^*}\right)_f + \left(\frac{1}{a}\right)_f \frac{4\mathbf{u}_f^n\left(\frac{V^n}{V^{n+1}}\right)_f - \mathbf{u}_f^{n-1}\left(\frac{V^{n-1}}{V^{n+1}}\right)_f}{2\Delta t}. \tag{25}$$

By substituting the expression for the cell-face velocity Eq. (23) into the discretized continuity equation Eq. (10), we obtain the discretized pressure equation which is of the same form as Eq. (12). Similarly, the update of the cell-face volume flux follows Eq. (13). However, it is crucial to observe that the definitions of $\left(\frac{1}{a}\right)_f$ and $\left(\frac{\mathbf{H}}{a}\right)_f$ have been changed.

The cell-face velocities from the previous time steps used in the last term on the RHS of Eq. (25) must satisfy the continuity equation at their corresponding time level. This is achieved by using [21, 26]

$$\mathbf{u}_f^n = \overline{(\mathbf{u}_P^n)}_f + \mathbf{n}_f^n \left[\frac{\dot{V}_f^n}{S_f^n} - \mathbf{n}_f^n \cdot \overline{(\mathbf{u}_P^n)}_f \right], \tag{26}$$

and a similar expression is used for $\mathbf{u}_f^{n-1}$.

The velocity of a moving boundary face-center $\mathbf{v}_s^{n+1}$ that is required in the discrete form of the GCL Eq. (3), is calculated using the first-order Euler scheme in the OpenFOAM version v2106. To obtain consistent temporal accuracy, the temporal discretization scheme for $\mathbf{v}_s^{n+1}$ should be consistent with the temporal discretization in the N-S equation. Following [26], we use

$$\mathbf{v}_s^{n+1} = \frac{3\mathbf{x}_s^{n+1} - 4\mathbf{x}_s^n + \mathbf{x}_s^{n-1}}{2\Delta t}, \tag{27}$$

where $\mathbf{x}_s$ is the position of the face center on the moving boundary.

2.3. Solution procedure

The complete iterative PISO solution procedure is shown in Algorithm 1. At the beginning of each time step, the mesh is updated, and the volume flow rate relative to the mesh motion is determined by solving a pressure correction equation. Subsequently, the momentum equation is assembled and solved to obtain a partial velocity and flow rate, which are then used in the pressure equation within a PISO step. The PISO loop involves iteratively updating the pressure and correcting the partial velocity based on the updated pressure. With the updated pressure, velocity, and flow rates, the momentum equation is assembled and solved again, followed by the PISO steps to satisfy continuity. The outer iterations continue until either the maximum iteration number is reached or the velocity residual falls below a specified tolerance. In OpenFOAM , the residual r for an arbitrary system of equations equation $A\psi = b$ is defined as $r = \frac{\|\mathbf{b}-\mathbf{A}\cdot\boldsymbol{\Psi}\|}{\|\mathbf{A}\cdot\boldsymbol{\Psi}-\mathbf{A}\cdot\overline{\boldsymbol{\Psi}}\|+\|\mathbf{b}-\mathbf{A}\cdot\overline{\boldsymbol{\Psi}}\|}$ [31], where $\|\dots\|$ is the matrix norm, and $\bar{\psi}$ is the mean value of ψ over all cells.

Algorithm 1: Iterative ALE PISO.

while $t^{n+1} < t_{end}$ **do**

  $\mathbf{u}^{n+1} = \mathbf{u}^n$, $p^{n+1} = p^n$, $\dot{V}_f^{n+1} = \dot{V}_f^n$;

  while $i < N_{outer}$ *or* $u_{tol} > \epsilon_u$ **do**

    if *i=0* **then**

      update the mesh;

      Get absolute volume flow rate $\dot{V}_f = \mathbf{u}_f \cdot \mathbf{n} S_f$;

      Solve $\nabla^2 p_{corr} = \nabla \cdot \dot{V}$;

      Correct Flux $\dot{V}_f = \dot{V}_f - \mathbf{n} \cdot (\nabla p)_n S_f$;

      Get relative volume flow rate $\dot{V} = \dot{V} - \dot{V}_{mesh}$;

    end

    Assemble and solve implicitly momentum equation for $\mathbf{u}^{n+1}$;

    while $j < N_{PISO}$ **do**

      Get partial velocity $\mathbf{u}_P^{n+1} = \frac{\mathbf{H}_P(\mathbf{u}^{n+1})}{a_P}$;

      Get partial volume flow rate $\dot{V}_f^{n+1} = \mathbf{n}^{n+1} \cdot \left(\frac{\mathbf{H}(\mathbf{u}^{n+1})}{a}\right)_f S_f^{n+1}$;

      while $k < N_{nonOrtho}$ **do**

        Assemble and solve pressure equation for p_P^{n+1} ;

      end

      Correct volume flow rate $\dot{V}_f^{n+1} = \dot{V}_f^{n+1} - \left(\frac{1}{a}\right)_f (\mathbf{n} \cdot \nabla p S)_f^{n+1}$;

      Correct velocity $\mathbf{u}_P^{n+1} = \mathbf{u}_P^{n+1} - \frac{1}{a_P} (\nabla p)_P^{n+1}$;

      j++;

    end

    i++;

  end

  $t^{n+1} = t^{n+1} + \Delta t$;

end

Please note that the first-order Euler scheme is used for the temporal discretization in Eq. (4) and Eq. (27) at the first time step since the values at $(n-1)$ are not available.

3. IBM solver

3.1. Governing equations

To simplify the presentation, we will assume that all moving boundaries are modeled with IBM. It should be noted that, using the ALE framework presented in section 2, one could easily create a hybrid ALE-IBM solver.

The continuity and momentum equations are now presented in a differential form to facilitate the derivation of the IBM body force calculation

$$\nabla \cdot \mathbf{u} = 0, \tag{28}$$

and

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}, \tag{29}$$

where $\mathbf{f}$ is the Eulerian body force vector resulting from distributing the IB force $\mathbf{F}$ defined on the immersed boundary Γ over a small volume of fluid around Γ. In continuous form, the Eulerian body force is given from the distributing operator $\mathcal{D}$ based on a regularized Dirac function $\delta(\mathbf{x})$ [32] as the kernel function

$$\mathbf{f} = \mathcal{D}\left[\mathbf{F}\right](\mathbf{x}) = \int \mathbf{F}(r)\delta(\mathbf{x} - \mathbf{X}(r))\, dV(r), \tag{30}$$

where r is a parameter used to quantify the Lagrangian quantities along the immersed boundary.

3.2. Body force calculation

Using the second-order backward scheme, the momentum equation (29) can be written in semi-discretized form

$$\frac{3\mathbf{u}^{n+1} - 4\mathbf{u}^{n} + \mathbf{u}^{n-1}}{2\Delta t} = C(\mathbf{u}^{n+1}, p^{n+1}) + \mathbf{f}^{n+1}, \tag{31}$$

where

$$C(\mathbf{u}, p) = -\mathbf{u} \cdot \nabla \mathbf{u} - \nabla p + \nu \nabla^2 \mathbf{u} \tag{32}$$

represents the convection, pressure gradient, and diffusion terms. Similar to the calculation of the non-linear convection term, we update the IBM body force in

the outer iterations. To update the force iteratively, we express the force at the new iteration using the velocity from the previous iteration i.e.

$$\mathbf{f}^{n+1,i+1} = \frac{3\mathbf{u}^{n+1,i} - 4\mathbf{u}^{n} + \mathbf{u}^{n-1}}{2\Delta t} - C(\mathbf{u}^{n+1,i}, p^{n+1,i}). \tag{33}$$

Introducing a new variable

$$\mathbf{u}^{*} = \frac{4\mathbf{u}^{n} - \mathbf{u}^{n-1} + 2C(\mathbf{u}^{n+1,i}, p^{n+1,i})\Delta t}{3}, \tag{34}$$

the force calculation Eq. (33) can be rewritten as

$$\mathbf{f}^{n+1,i+1} = \frac{3}{2}\frac{\mathbf{u}^{n+1,i} - \mathbf{u}^{*}}{\Delta t}. \tag{35}$$

To accurately impose boundary conditions on the immersed boundary, the fluid velocity at the Lagrangian marker $\mathbf{u}^{n+1}(\mathbf{X}_l)$ must equal the velocity of the marker (indicated by a subscript l) itself $\mathbf{V}_l^{n+1}$ (which is known from the prescribed boundary motion)

$$\mathbf{u}^{n+1}(\mathbf{X}_l) = \mathbf{V}_l^{n+1}. \tag{36}$$

The fluid velocity at the Lagrangian marker is obtained by interpolation of the fluid velocity $\mathbf{u}^{n+1,i}$ from cell centers near the marker position

$$\mathbf{u}^{n+1}(\mathbf{X}_l) = \mathcal{I}[\mathbf{u}^{n+1,i}](\mathbf{X}_l). \tag{37}$$

The interpolation operator is defined as

$$\mathcal{I}[\phi](\mathbf{X}_l) = \sum_{i=1}^{N_l} \phi(\mathbf{x}_i)\delta(\mathbf{x}_i - \mathbf{X}_l)\Delta v_i, \tag{38}$$

where X_l is the position of the Lagrangian marker, and x_i's ($i = 1 \cdots N_l$), are the positions of the Eulerian cells used to interpolate $\phi(x_i)$ to get $\phi(X_i)$, and Δv_i is the cell volume of the fluid. The kernel function is expressed as $\delta_h(\mathbf{r}) = \delta_h^{1D}(r_x)\delta_h^{1D}(r_y)\delta_h^{1D}(r_z)$ with $r = (r_x, r_y, r_z) = \frac{|\mathbf{x}-\mathbf{X}|}{h}$, where h is the local Eulerian mesh size. In this study, the regularized delta function proposed by Roam et. al[33] is adopted

$$\delta_h^{1D}(d) = \begin{cases} \frac{1}{6h}\left[5 - 3d - \sqrt{-3(1-d^2)+1}\right], & 0.5 \le d \le 1.5 \\ \frac{1}{3h}(1 + \sqrt{-3d^2+1}), & d \le 0.5 \\ 0, & \text{othersize.} \end{cases} \tag{39}$$

If we apply the interpolation operator to Eq. (35) and further use Eq. (36) and Eq. (37), we get the IB force at the Lagrangian marker

$$\mathbf{F}^{n+1,i+1} = \frac{3}{2}\frac{\mathbf{V}^{n+1} - \mathcal{I}(\mathbf{u}^*)}{\Delta t}. \tag{40}$$

To get the desired Eulerian body force, we distribute the Lagrangian body force to the Eulerian grid points

$$\mathbf{f}^{n+1,i+1} = \mathcal{D}[\mathbf{F}^{n+1,i+1}] = \frac{3}{2}\mathcal{D}\left[\frac{\mathbf{V}^{n+1} - \mathcal{I}(\mathbf{u}^*)}{\Delta t}\right]. \tag{41}$$

Here, $\mathcal{D}[\mathbf{F}^{n+1,i+1}]$ is a discrete version of the distribution operator Eq. (30) defined using the same kernel function as used for the interpolation. Specifically, the body force of an Eulerian cell centered at $\mathbf{x}_i$ can be computed from its surrounding Lagrangian markers $l = 1, ..., N$ as

$$\mathbf{f}^{n+1,i+1}(\mathbf{x}_i) = \sum_{l=1}^{N} \mathbf{F}_l^{n+1,i+1} \delta_h(\mathbf{X}_l - \mathbf{x}_i) \Delta V_l, \tag{42}$$

where $\mathbf{F}_l^{n+1,i+1}$ is the body force of the l-th surrounding Lagrangian marker. The associated volume ΔV_l is calculated as $\Delta V_l = \Delta S_l h$, where ΔS_l is the control surface area of marker l, and h is taken as the Eulerian grid spacing $h = \Delta x = \Delta y$ for a uniform Cartesian grid. The Eulerian force $\mathbf{f}^{n+1,i+1}$ is then used to solve for the velocity $\mathbf{u}^{n+1,i+1}$ in the next iteration.

Several researchers [12, 19, 32] have shown that the direct forcing IBM for explicit time discretization leads to a "velocity slip" $\mathcal{I}[\mathbf{u}^{n+1}](\mathbf{X}_l) \neq \mathbf{V}_l^{n+1}$. We show that this is also the case for the implicit time discretization adopted here. After the convergence of the outer iterations we have e.g. $\mathbf{f}^{n+1,i+1} = \mathbf{f}^{n+1,i} = \mathbf{f}^{n+1}$. Applying the interpolation operator to Eq. (35) we find (dropping the position of the Lagrangian marker $\mathbf{X}_l$ for brevity)

$$\mathcal{I}\left[\mathbf{u}^*\right] = \mathcal{I}\left[\mathbf{u}^{n+1}\right] - \frac{2}{3}\Delta t \mathcal{I}\left[\mathcal{D}[\mathbf{F}^{n+1}]\right]. \tag{43}$$

Substituting this into equation Eq. (40), we find the slip error

$$\mathbf{V}^{n+1} - \mathcal{I}\left[\mathbf{u}^{n+1}\right] = \frac{2}{3}\Delta t \left(\mathbf{F}^{n+1} - \mathcal{I}\left[\mathcal{D}[\mathbf{F}^{n+1}]\right]\right). \tag{44}$$

Gsell et al. [12] have shown that for regularized Delta functions as kernel functions one can estimate $\mathcal{I}\left[\mathcal{D}[\mathbf{F}^{n+1}]\right] \approx \kappa \mathbf{F}^{n+1}$, where κ depends on the specific regularized Delta functions used for interpolation and distribution operators. The boundary slip error only vanishes if the velocity can be reproduced under the composition of the interpolation and the distribution operators, i.e. $\kappa = 1$, but for most kernel functions $\kappa < 1$ [12, 32] and thus a substantial boundary slip error is observed. Two previously proposed strategies to mitigate the slip error will be presented in section 3.3 below and their effect on the overall IBM accuracy is discussed in section 6.1.

At the first time step, no previous time step solution $\mathbf{u}^{n-1}$ is available and thus we need to use a first-order implicit Euler scheme $\frac{d\mathbf{u}}{dt} \approx \frac{\mathbf{u}^{n+1,i}-\mathbf{u}^{n}}{\Delta t}$. Following a similar derivation as shown above, we find the expression for the force calculation to be

$$\mathbf{f}^{n+1,i+1} = \mathcal{D}\left[\frac{\mathbf{V}^{n+1} - I(\mathbf{u}^{*})}{\Delta t}\right], \tag{45}$$

and the corresponding $\mathbf{u}^{*}$ expression simplifies to

$$\mathbf{u}^{*} = \mathbf{u}^{n} + C(\mathbf{u}^{n+1,i}, p^{n+1,i})\Delta t. \tag{46}$$

3.3. Body force correction

The velocity slip defined in equation (44) causes a non-physical flow penetration across the solid boundary. To reduce the slip error, different approaches have been proposed for the body force calculation. Here, we compare two strategies: the iterative method proposed by Ji et al. [19] and a correction factor method proposed by Gsell and Favier [32].

In Ji's iterative method, a separate iteration is used to correct the body force. The backward Euler method in Eq. (41) can be written in an iterative form, as shown in Eq. (47). In this study, the relaxation coefficients α and β are both set to 1. When the iteration converges, the second term becomes 0, and $\mathbf{f}^{n+1,i+1,k+1}$ equals $\mathbf{f}^{n+1,i+1,k}$, based on Eq. (35).

$$\mathbf{f}^{n+1,i+1,k+1} = \alpha \mathbf{f}^{n+1,i+1,k} + \beta \frac{3}{2}\mathcal{D}\left[\frac{\mathbf{V}^{n+1} - \mathcal{I}(\mathbf{u}^{*}) - \frac{2}{3}\mathbf{f}^{n+1,i+1,k}}{\Delta t}\right]. \tag{47}$$

The algorithm for the iterative force correction is shown in Fig. 2. Ji et al. [19] conducted tests and found that in most cases, ten cycles (nINNER=10 in Algorithm 2) work well, but they also observed that for some extreme cases, it may not be enough to achieve a convergent solution. Therefore, the number of inner loop cycles can be somewhat arbitrary.

Algorithm 2: Iterative Force Correction for Backward Scheme

$\mathbf{f}^0 = \mathbf{0}$;

while *($k < nINNER$)* **do**

  $\mathbf{u}^* = \frac{1}{3}\left[4\mathbf{u}^n - \mathbf{u}^{n-1} + 2C(\mathbf{u}^{n+1}, p^{n+1})\Delta t\right] + \frac{2}{3}\mathbf{f}^k\Delta t$;

  Interpolate the flow velocity to the Lagrangian markers $\mathbf{u}_l = \mathcal{I}(\mathbf{u}^*)$;

  Get the body forces at the Lagrangian markers $\mathbf{f}_l = \frac{\mathbf{V}^{n+1} - \mathbf{u}_l}{\Delta t}$;

  Spread the Lagrangian force to the fluid cells $\mathbf{f}^{k+1} = \mathbf{f}^k + \frac{3}{2}\mathcal{D}(\mathbf{f}_l)$;

  k++;

end

$\mathbf{f}^{n+1} = \mathbf{f}^k$;

Gsell et al. [12, 32] introduced a simpler method for correcting immersed forces. Their approach is based on using constant correction factors for different delta functions to address the boundary slip error caused by the non-reciprocity of the interpolation $\mathcal{I}$ and distribution operators $\mathcal{D}$ in the direct-forcing immersed boundary method. With this method, the force obtained from Eq.(41) is corrected by a factor γ

$$\mathbf{f}^{n+1,i+1} = \frac{3}{2\gamma}\mathcal{D}\left[\frac{\mathbf{V}^{n+1} - \mathcal{I}(\mathbf{u}^*)}{\Delta t}\right]. \tag{48}$$

The constant factor γ depends only on the regularized delta functions used for interpolation and distribution. Gsell and Favier [32] suggested a factor of $\gamma = 0.5$ for the delta function used in this work (i.e. equation (39)).

3.4. IBM PISO with Rhie-Chow interpolation

A key result of Tuković et al. [21] was the development of a consistent (i.e., independent of the time step) RC interpolation method for the second-order

backward ALE solver. We have presented the key steps in section 2.2 and focus here only on the specific RC interpolation aspects of the IBM solver.

The main difference between the IBM and ALE solvers is the presence of a strongly varying (essentially discontinuous) body force in the former. For such cases, it has been observed that the flow field might exhibit undesired oscillations even with the RC method, as noted by Zhang et al. [34]. Thus, an appropriate RC interpolation method for the body force becomes crucial [29, 34]. We follow the ideas of [29, 35, 36] by using cell-center values that are reconstructed from face-center values for the body force and pressure gradient in the momentum predictor. These reconstructed fields can be shown to be smoother than the original cell-centered variables [36]. The reconstruction method of Weller and Shahrokhi [35] for a cell-center vector field $\mathbf{v}$ (either body force or pressure gradient) is given by

$$\mathbf{v}_{\text{recons}} = \left(\sum_f \mathbf{S}_f \mathbf{n}_f \right)^{-1} \cdot \left[\sum_f (\mathbf{v}_f \cdot \mathbf{S}_f) \mathbf{n}_f \right], \tag{49}$$

where $\mathbf{v}_f$ is obtained by linear interpolation from the cell center values. The reconstruction method (49) and the linear interpolation to face centers can be viewed as approximate inverse operations. It should be noted that the reconstruction approach is also used in standard OpenFOAM solvers for problems with buoyancy force terms (*buoyantPimpleFoam*) and flows with interface tracking (*interFoam*).

The $\mathbf{H}$ operator is now defined as

$$\mathbf{H}(\mathbf{u}_P) = a_P \mathbf{u}_P + (\nabla p)_{recons} - \mathbf{f}_{recons}, \tag{50}$$

and it does not include the contributions from the body force or the pressure gradient. The explicit expression for cell center velocity $\mathbf{u}_P$ is given by

$$\mathbf{u}_P = \frac{\mathbf{H}(\mathbf{u}_P)}{a_P} - \frac{(\nabla p)_{recons}}{a_P} + \frac{\mathbf{f}_{recons}}{a_P}, \tag{51}$$

and this equation will be used to update the cell center velocity after the pressure equation has been solved. The cell-face velocity $\mathbf{u}_f$ can be obtained in the spirit

of RC by mimicking the momentum equation on the face

$$\mathbf{u}_f = \left[\frac{\mathbf{H}(\mathbf{u}_P)}{a}\right]_f - \left(\frac{1}{a}\right)_f (\nabla p)_f + \left(\frac{1}{a}\right)_f \mathbf{f}_f. \tag{52}$$

Here the small stencil pressure face-gradient $(\nabla p)_f$ and the body force f_f interpolated to the face center are used instead of the "interpolated reconstructed variables" because of the approximate inverse relationship mentioned above. Note the same definitions as in section 2.2 are used for $\left[\frac{\mathbf{H}(\mathbf{u}_P)}{a}\right]_f$ and $\left(\frac{1}{a}\right)_f$. The small stencil used for the face-center pressure gradient $(\nabla p)_f$ prevents the decoupling of the pressure and the velocity.

Substituting the cell-face velocity $\mathbf{u}_f$ obtained from Eq. (52) into the discretized continuity equation (10) yields the discrete pressure equation including the RC interpolation of the body force

$$\begin{aligned}\sum_f \left(\frac{1}{a}\right)_f \mathbf{n}_f^{n+1} \cdot (\nabla p)_f^{n+1} S_f^{n+1} = \sum_f \mathbf{n}_f^{n+1} \cdot \left(\frac{\mathbf{H}}{a}\right)_f S_f^{n+1} \\ + \sum_f \mathbf{n}_f^{n+1} \cdot \left(\frac{1}{a}\right)_f \mathbf{f}_f S_f^{n+1}.\end{aligned} \tag{53}$$

Once the pressure equation is solved, the cell-face volume flow rate that satisfies the continuity equation can be calculated by

$$\dot{V}_f^{n+1} = \mathbf{n} \cdot \left[\frac{\mathbf{H}(\mathbf{u}_P)}{a_P}\right]_f S_f - \left(\frac{1}{a_P}\right)_f (\mathbf{n} \cdot \nabla p)_f^{n+1} S_f + \left(\frac{1}{a_P}\right)_f (\mathbf{n} \cdot \mathbf{f})_f^{n+1} S_f, \tag{54}$$

and the cell-center velocity is then updated from the predicted velocity with the reconstructed pressure gradient and body force through

$$\mathbf{u}_P^{n+1} = \frac{\mathbf{H}(\mathbf{u}_P)}{a_p} - \frac{1}{a_P}(\nabla p)_{recons}^{n+1} + \frac{1}{a_P}\mathbf{f}_{recons}^{n+1}. \tag{55}$$

3.5. Iterative PISO-based IBM

The iterative PISO algorithm with IBM can be summarized as follows (Algorithm 3). At the time step $(n+1)$, the velocity, pressure, and body force are initialized using their values from the previous time step (n). The momentum equation is then solved with the reconstructed pressure gradient and body force, followed by the pressure equation, which corrects the velocity in the PISO loop.

In cases where the mesh is non-orthogonal, a non-orthogonality correction is applied during the solution of the pressure equation. The updated velocity is then used to calculate the body force. This process is repeated for each outer loop, with the latest information on velocity, pressure, and body force, until the outer iteration converges. It is worth noting that the presented IBM solver has the ability to handle some boundaries using dynamic mesh techniques, but in this study, a static mesh is used for all IBM simulations.

4. Verification of the solvers

We established the correctness of the implementation of the ALE and IBM solver through a temporal convergence study using the same test cases as those in Tuković et al. [21] to verify the expected second-order temporal accuracy. We consider a sequence of simulations using increasingly small time steps and take the results obtained with the smallest time step as the "exact" solution for the error calculation.

For both ALE and IBM solvers, we use the second-order upwind scheme (referred to as *linearUpwind* in OpenFOAM) for the spatial discretization of the convection term. For the Laplacian terms (diffusion and pressure equation), the Gauss divergence theorem is used to convert the volume integral to the surface integral, and the surface normal gradient is discretized with a tight stencil using the cell values straddling the face and including an explicit non-orthogonal correction (*Gauss linear corrected* in OpenFOAM).

The cell center gradients of velocity and pressure are computed using the least squares scheme and Gauss linear scheme (employing the Gauss divergence theorem), respectively. Linear interpolation is applied for all variables to obtain face values from cell values.

The Rhie-Chow interpolation is adopted for the pressure in the ALE solver, and for the pressure and body force in the IBM solver. For the IBM solver, the correction factor method (Eq. (48) is adopted unless stated otherwise.

A convergence criterion of $\epsilon_1 = 10^{-8}$ is used for the absolute velocity residual

Algorithm 3: Iterative IBM PISO

while $t^{n+1} < t_{end}$ **do**

$\mathbf{u}^{n+1} = \mathbf{u}^n$, $p^{n+1} = p^n$, $\dot{V}_f^{n+1} = \dot{V}_f^n$ $\mathbf{f}^{n+1} = \mathbf{f}^n$; **while** $i < N_{outer}$ *or* $u_{tol} > \epsilon_u$ **do**

Assemble and solve implicitly momentum equation for $\mathbf{u}^{n+1}$ with the reconstructed pressure gradient and the reconstructed body force;

while $j < N_{PISO}$ **do**

Get partial velocity $\mathbf{u}_P^{n+1} = \frac{\mathbf{H}_P(\mathbf{u}^{n+1})}{a_P}$;

Get partial volume flow rate with the linear interpolated body force on the face $\dot{V}_f^{n+1} = \mathbf{n}^{n+1} \cdot \left[\left(\frac{\mathbf{H}(\mathbf{u}^{n+1})}{a} \right)_f + \overline{\mathbf{f}}^{n+1} \right] S_f^{n+1}$;

while $k < N_{nonOrtho}$ **do**

Assemble and solve pressure equation for p^{n+1};

end

Correct volume flow rate $\dot{V}_f^{n+1} = \dot{V}_f^{n+1} - \left(\frac{1}{a}\right)_f (\mathbf{n} \cdot \nabla p S)_f^{n+1}$;

Correct velocity $\mathbf{u}_P^{n+1} = \mathbf{u}_P^{n+1} - \frac{1}{a_P} \left(\frac{\dot{V}_f^{n+1} - \mathbf{n}^{n+1} \cdot \overline{\mathbf{f}^{n+1}} S_f^{n+1}}{a_f} \right)_{recons}$;

j++;

end

Update the body force $\mathbf{f}^{n+1}$ using the correction-factor or the iteration method ;

i++ ;

end

$t^{n+1} = t^{n+1} + \Delta t$;

end

to exit the outer iterations. The convergence criteria for the linear equation solvers of velocity, pressure, and mesh motion are all set to $\epsilon_2 = 10^{-15}$.

For cases with non-orthogonal meshes, we reduce the error arising from the

non-orthogonality with a deferred-correction technique [29, 37] for the pressure using one correction step. Two inner (PISO) corrector steps are utilized per outer iteration (for a total of four pressure solves per outer iteration). Further details about the OpenFOAM schemes can be found in the works of Darwish and Moukalled [29] and Jasak [38].

For the static mesh cases, the ALE solver simply becomes a conventional spatially and temporally second-order accurate viscous flow solver. For the other cases, the mesh motion is sufficiently small such that the mesh quality remains high without re-meshing.

4.1. ALE solver

Tuković et al. [21] demonstrated second-order temporal convergence of the ALE solver and we verified our implementation using the same test cases. Note that we will not compare the errors obtained with our implementation with those from [21] since they are based on different OpenFOAM versions, slightly different spatial discretization schemes, and different grids (for the two cylinder cases).

The first case is the lid-driven cavity flow [21] with $\mathrm{Re} = VL/\nu = 10$, where $V = 1$ m/s is the velocity of the top wall and $L = 1$ m is the length of the square domain which is discretized into 50×50 uniform cells. Using the time scale $T = L/V = 1$ s, the simulations start from the fluid at rest and run to the final time $t_{\mathrm{final}}/\mathrm{T} = 0.05$, at which the maximal errors (L_∞ norm) for different time-step sizes are evaluated for the velocity and pressure fields ($\epsilon_v = \|v - v_{\mathrm{exact}}\|_\infty$, and $\epsilon_p = \|p - p_{\mathrm{exact}}\|_\infty$). The temporally "exact" solution is taken to be the numerical solution obtained with a small time-step size of $t_{\mathrm{fine}}/T = 1 \times 10^{-5}$. This test case is only considered for the ALE solver.

Fig. 1 shows the ALE solver is second-order accurate in time for both velocity and pressure. Following [21], we did the same temporal study for the standard non-iterative OpenFOAM solver (called *icoFoam*) and also found that *icoFoam* is no better than first-order even though the same second-order accurate backward scheme was used.

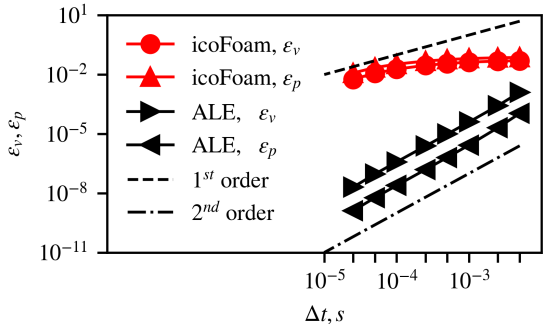

Figure 1: Temporal convergence rate of the ALE solver for the lid-driven cavity flow.

The next two test cases consider the flow around a fixed and oscillating cylinder in a confined channel. The computational domain spans $[-2D, 20D] \times [2.1D, 2D]$ in stream-wise (x) and cross-stream (y) directions, respectively. The center of the cylinder is located at $x = y = 0$ and the domain is discretized using a grid consisting of 7,640 quadrilateral cells as shown in Fig. 2.

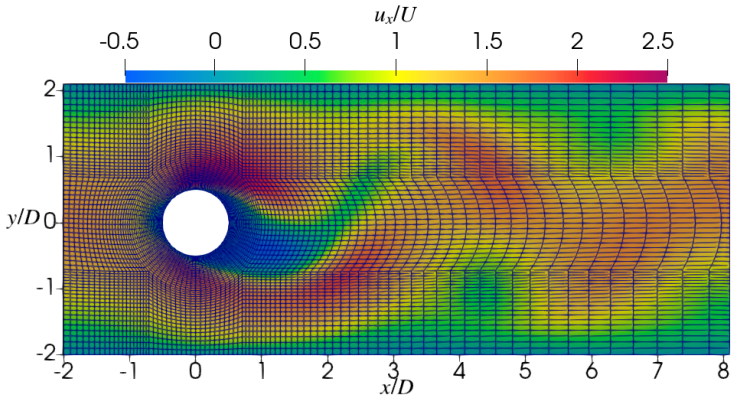

Figure 2: Inlet portion of the grid for the confined cylinder cases used by ALE solver with velocity contours.

The Reynolds number is set to $\mathrm{Re} = UD/\nu = 100$, following the setup in [21]. Here, U represents the average velocity of the parabolic inlet velocity profile, set to $U = 1$ m/s, and the cylinder diameter is $D = 1$ m.

At the walls and cylinder, a no-slip boundary condition is adopted and at the outlet, we prescribe a zero-gradient for velocity and a fixed value for pressure.

The asymmetry of the flow promotes the transition to a periodic state [21, 30] and thus reduces the computational time. The vortex shedding period for this case is $T_s = 3.3$ s.

In the oscillating cylinder case, the cylinder undergoes oscillations in cross-stream direction with a position given by $y = \frac{1}{4} D \sin(2\pi f t)$ with an oscillation frequency of $f = 0.3$ Hz (or a period of $T_o = 1/f = 3.3$ s which is equal to the static cylinder vortex shedding period $T_s = T_o = T$). All other parameters are the same as in the fixed cylinder case.

Both cases are run for $t_f/T = 18$ which means about 10 vortex shedding periods are simulated after reaching the fully periodic state. The error of the lift C_l and drag C_d coefficients are calculated at the final time using the smallest time step simulation results (with $\Delta t/T = 3 \times 10^{-5}$) as the true solution.

The temporal convergence of the lift and drag coefficient errors for the static cylinder is shown in Fig. 3 (left) which clearly shows the second-order convergence. For the oscillating cylinder (right), the convergence behavior is second-order beyond the largest time-step result which had a maximum CFL number around $CFL_{max} \approx 1.6 > 1$.

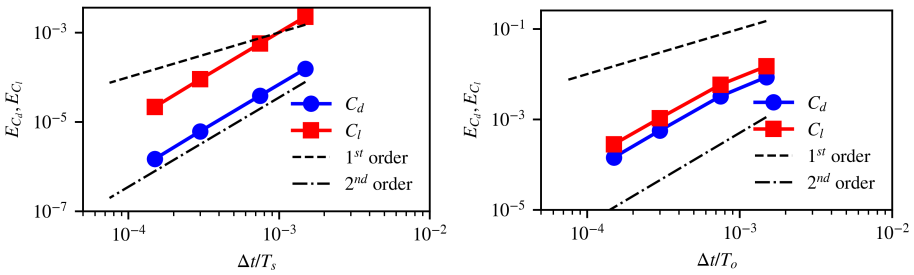

Figure 3: Temporal convergence rate of the ALE solver for flow past a static cylinder (left) and oscillating cylinder (right) in a confined channel.

4.2. IBM solver

The IBM solver was verified using the same cylinder flow cases as used for the ALE solver verification shown in section 4.1. However, a different mesh was used since the IBM method requires a uniform grid around the immersed boundary. The mesh is shown in Fig. 4 and consists of a structured block located in the

region around the cylinder ($[-2D, 2D] \times [-D, D]$) and a mixture of structured and unstructured regions away from the cylinder with a total of 21,468 cells. Previous research [3, 17] suggests that the distance between Lagrangian markers should be small enough to avoid leakage, but numerical instability can arise if markers are too close. An optimal spacing ratio of order 1 is recommended. Based on our experience, the optimal configuration for the IBM solver is to have approximately 1-2 markers per fluid cell. For the convergence study presented in this work, we prioritize accuracy over computational cost and use a ratio of 4 for the fluid grid spacing and Lagrangian marker distance.

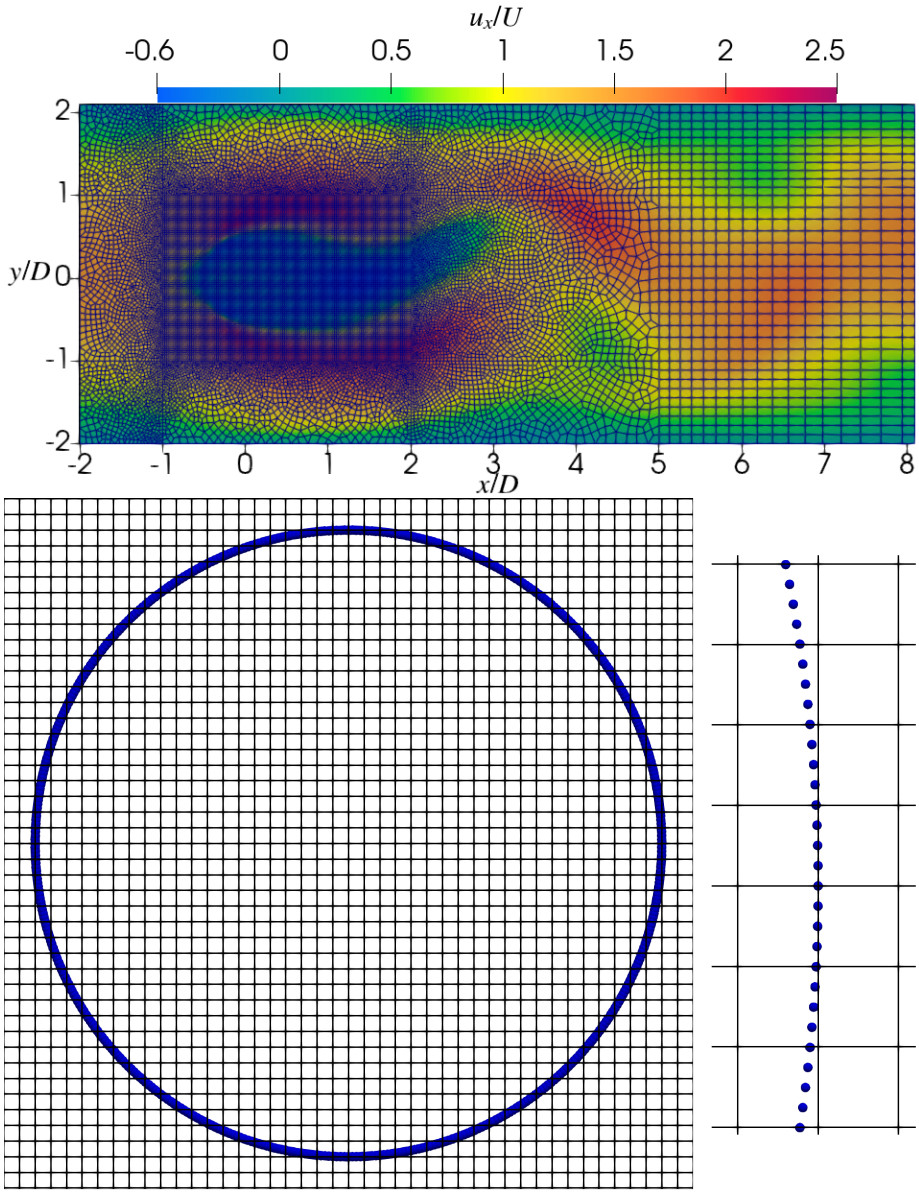

Figure 4: Inflow region of the unstructured grid (top) for the confined cylinder cases used by the IBM solver and a close-up of the cylinder region with Lagrangian marker distribution (bottom).

Only the cylinder surface was treated by IBM whereas the top and bottom walls use no-slip boundary conditions as was used for the ALE solver. To

calculate the lift and drag coefficients, the total force exerted by the fluid on the immersed boundary can be obtained through volume integration of the volumetric Eulerian body force or the volumetric Lagrangian body force and the inertial force if the immersed boundary is moving, given by:

$$\mathbf{F}_{total} = -\sum_i \mathbf{f}_i v_i + F_{inertia} = \sum_j \mathbf{F}_j V_j + F_{inertia} \tag{56}$$

Here, v_i and V_j represent the volume of fluid cell i and marker j, respectively; $\mathbf{f}_i$ and $\mathbf{F}_j$ are the Eulerian body force acting on cell i and Lagrangian body force from the marker j, respectively. The negative sign indicates that the fluid exerts a force on the immersed boundary. $F_{inertial}$ is the inertial force induced by the boundary motion. It is important to note that $\mathbf{f}_i$ is nonzero only if cell i is in the immediate neighborhood of a Lagrangian marker. Additionally, as noted by other researchers [39, 40], neglecting the inertial force can result in overestimating the drag coefficient. The error of the lift C_l and drag C_d coefficients are calculated at the final time using the smallest time step simulation results (with $\Delta t/T = 3 \times 10^{-5}$) as the true solution.

Fig. 5 shows the temporal convergence of the errors for the static cylinder (left) and oscillating cylinder (right). Overall, the IBM solver exhibits a second-order temporal convergence rate in both cases except for the largest time-step results that have $CFL_{max} \approx 1.6$. Although the velocity-slip error arising from equation (44) is proportional to the time-step, the adopted force correction method is capable of sufficiently reducing the slip error for all but the largest time-step case such that the second-order convergence is observed. It is important to highlight that even though the algorithm is not strictly second-order accurate for the larger time steps, it remains stable even for CFL_{max} up to 8 (not shown here).

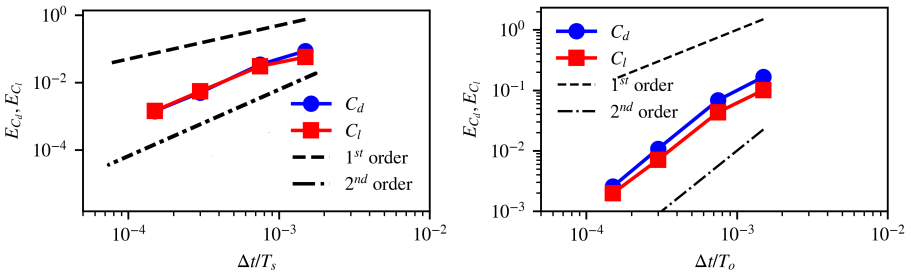

Figure 5: Temporal convergence rate of the IBM solver for flow past a static cylinder (left) and oscillating cylinder (right) in a confined channel.

5. Validation of the IBM solver

In the previous section, we demonstrated the expected second-order temporal convergence of the ALE and IBM solvers. In this section, we validate the IBM solver against the ALE solver for static and moving mesh cases. The ALE solver results are considered the ground truth since they are based on using a no-slip boundary condition on a body-conforming mesh with sufficient resolution in the boundary layer. In IBM, on the other hand, the no-slip boundary condition is modeled with the use of body forces. Three test cases are considered for the IBM solver validation: the flow past a stationary circular cylinder in a large domain, the flow around a stream-wise oscillating cylinder, and the flow around a cross-stream oscillating cylinder. All three cases are commonly used to test IBM solvers.

We want to ensure that any observed difference between the ALE and IBM results is solely due to the use of IBM and thus all other aspects of the numerical solution must be kept identical. This is achieved by using the same spatial discretization methods, the same linear equation solvers, the same boundary conditions (except for the solid boundary), and the same computational mesh. The latter can only be achieved approximately and we thus state that the meshes used for both solvers should be as similar to each other as possible. We achieve this by first designing the IBM mesh with a structured mesh region around the cylinder and then cutting out the cylinder with a few layers of unstructured

cells that conform to the cylinder surface as shown in Fig. 6.

For both ALE and IBM solvers, we utilize the same schemes and solver settings as stated in the verification section 4, except for the spatial discretization of the convection term which is changed to a second-order central scheme (referred to as *Gauss linear* in OpenFOAM) to also facilitate a comparison with some previous studies.

5.1. Flow past a stationary circular cylinder with uniform inflow

The laminar flow around a stationary circular cylinder in a large two-dimensional domain is the most commonly used benchmark problem to validate IBM solvers. Three cases were conducted using Reynolds numbers of 40, 100, and 185, which are based on the uniform inlet velocity U and the cylinder diameter D. The cylinder remains fixed at the origin, and adequate domains are adopted to eliminate the influence of the outer boundary on the study area near the cylinder, as previous and present studies suggest. The simulations continue until either the flow reaches a steady state (for Re=40) or 20 vortex shedding periods have passed (for Re=100 and 185), with a nominal CFL number $CFL = U\Delta t/\Delta x = 0.2$ where Δx is the grid spacing of the structured mesh.

5.1.1. Steady-state Re = 40 case

Gautier et al. [41] studied this case with a pseudo-spectral solver using a large domain and a very fine grid spacing to provide a benchmark solution. The low Reynolds number leads to a steady symmetric wake which makes this a very simple test case for the validation of the ALE and IBM solvers. We consider a truncated computational domain $[-10D, 10D] \times [-10D, 10D]$ discretized into a 1440×1440 uniform Cartesian grid for the IBM solver. The mesh spacing is the same as in [41]. The IBM solver used 500 Lagrangian markers uniformly distributed in the circumferential direction around the cylinder. The corresponding ratio between the fluid grid spacing and the Lagrangian marker distance is approximately 2.

To ensure a fair comparison between the ALE and the IBM solver, the meshes used for both solvers should be as similar as possible. To achieve this, the cylinder was extracted from the uniform background mesh used for IBM with the help of the *snappyHexmesh* tool which was used without adding refinement layers. The conformal mesh for the ALE solver is shown in Figure 6.

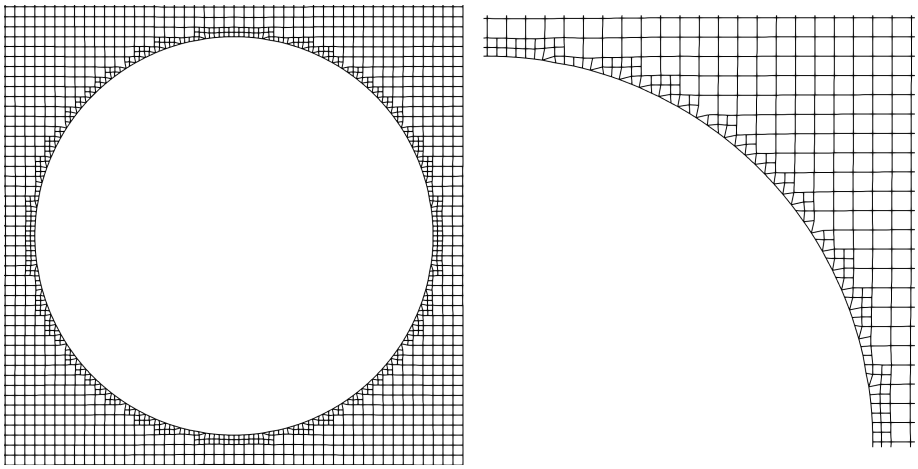

Figure 6: The mesh design for ALE solver near the cylinder (right panel shows the zoomed-in of the first quadrant of the cylinder).

To eliminate errors introduced by the truncated domain compared to [41], the velocity at the inlet, bottom, and top boundary is specified directly from the reference solution of [41]. At the outlet, we use a Dirichlet BC for pressure and Neumann BC for velocity.

The drag coefficient obtained from our ALE solver on the truncated domain is $C_{d,ALE} = 1.498$ which is very close to the value found by [41] $C_{d,SPECT} = 1.49$. Gautier et al. [41] compared the drag coefficient values from more than 15 publications (all based on no-slip BC on the cylinder) which vary between $C_d = 1.48 - 1.62$ thus demonstrating the accuracy of our ALE solver.

We found the IBM solver to be very accurate when compared to the ALE solver for this low Reynolds number steady-state problem with a drag coefficient of $C_{d,IBM} = 1.505$ and a relative error of only 0.47% (calculated based on our

ALE result).

5.1.2. Unsteady cases with Re = 100, and 185

For the cases with Reynolds numbers of 100 and 185, a larger computational domain of size $[-20D, 40D] \times [-20D, 40D]$ was utilized. The IBM solver employed a uniform Cartesian grid with a grid size of $D/50$ in the zone of $[-2D, 6D] \times [-2D, 2D]$ which encloses the cylinder. The mesh spacing in the remaining zones was gradually increased. In the IBM simulation, 320 Lagrangian markers are uniformly distributed in the circumferential direction around the cylinder; the corresponding ratio between the fluid grid spacing and the Lagrangian marker distance is 2.. The uniform velocity profile was prescribed at the inlet for both cases and the kinematic viscosity was adjusted to achieve the desired Reynolds number. A free-slip (i.e. no-stress) boundary condition was set at the top and bottom boundaries. At the outlet, we use a Dirichlet BC for pressure and Neumann BC for velocity.

Table 1 compares the ALE and IBM solver results for drag coefficient, lift coefficient, and Strouhal number $St = \dfrac{fD}{U}$. The results of IBM are in good agreement with those of ALE, with the C_d error increasing with Reynolds numbers but remaining below 3.5% for both cases. Additionally, IBM accurately predicts the lift coefficient and Strouhal number.

Table 1: Drag coefficient, lift coefficient, and Strouhal number comparisons between IBM and ALE for the flow past a stationary circular cylinder in a large domain with the uniform inflow at Re=100 and Re =185.

Re	IBM			ALE			E_{C_d}
	C_d	$C_{l,rms}$	S_t	C_d	$C_{l,rms}$	S_t	
100	1.379 ± 0.011	0.330	0.165	1.348 ± 0.011	0.327	0.166	2.30%
185	1.383 ± 0.055	0.458	0.192	1.340 ± 0.041	0.447	0.192	3.21%

5.2. Circular cylinder oscillating in quiescent flow

Many studies of fluid and structure interaction using IBM [42–45] consider an oscillating cylinder in a quiescent fluid as studied experimentally by DÜTSCH et al. [46]. This flow is characterized by two dimensionless parameters: the Reynolds number, defined as $Re = U_{max}D/\nu$, and the Keulegan-Carpenter number, defined as $KC = U_{max}/fD$, where U_{max} is the maximum velocity of the cylinder motion, and f is the oscillation frequency. Some researchers use the Stokes number, which is defined as the ratio of the Reynolds number and the Keulegan-Carpenter number, $\beta = Re/KC$, as an alternative characteristic parameter.

The harmonic oscillation motion of the cylinder is prescribed by the trajectory of its center as

$$X(t) = -A\sin(2\pi ft), \tag{57}$$

and correspondingly of the velocity of its center

$$U(t) = \dot{X}(t) = -2\pi fA\cos(2\pi ft), \tag{58}$$

where A denotes the amplitude of the motion. Thus, we have $Re = 2\pi fAD/\nu$ and $KC = 2\pi A/D$.

For our validation, we consider the following parameters: $Re = 100$ and $KC = 5$. A large computational domain with an extension of $55D \times 35D$ is considered as shown in Fig. 7. The cylinder is initially placed at the center of the domain. A non-uniform Cartesian grid of 250×200 cells is used to discretize the entire domain, with the region of $-D \leq x, y \leq D$ being refined (with a uniform grid spacing of $D/40$) to improve accuracy (see Fig. 7). The conforming mesh for the ALE solver is again created using the *snappyHexmesh* tool as described in section 5.1.1. In the IBM simulation, 800 Lagrangian markers are uniformly distributed in the circumferential direction around the cylinder; the corresponding ratio between the fluid grid spacing and the Lagrangian marker distance is 1.0. This is a confined flow case with no-slip walls prescribed at all the lateral boundaries. The non-dimensional time step, $t = t/T$, where $T = 1/f$

is the oscillating period, is set to 1/720, and the maximum CFL number remains below 0.5.

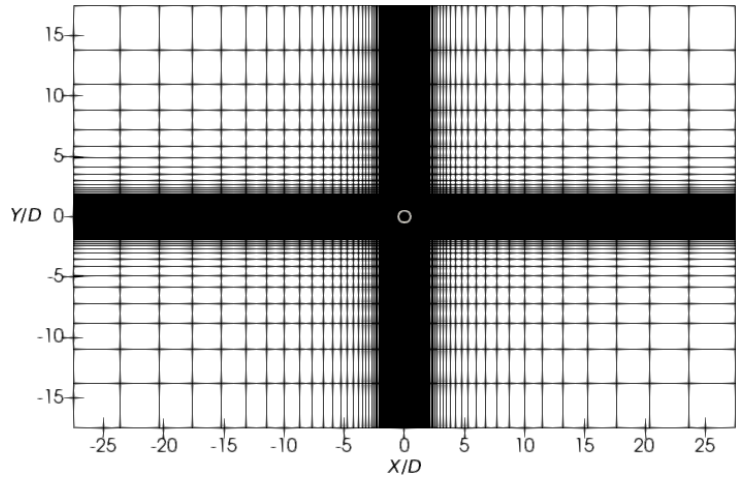

Figure 7: The mesh design for IBM solver near the in-line oscillating cylinder

The simulations are initialized with a stationary flow field and the cylinder undergoes harmonic motion until periodic vortex shedding occurs. Fig. 8 shows the comparison of the pressure and vorticity contours from the IBM and the ALE results at four different phase angles after vortex shedding commences. The top half of each contour plot shows the results from the IBM solver, while the bottom half shows the results from the ALE solver (the flow field is symmetric about the horizontal axis of the cylinder). The IBM and ALE results are very similar with almost symmetrical contours formed by both solvers at different phase angles, as expected.

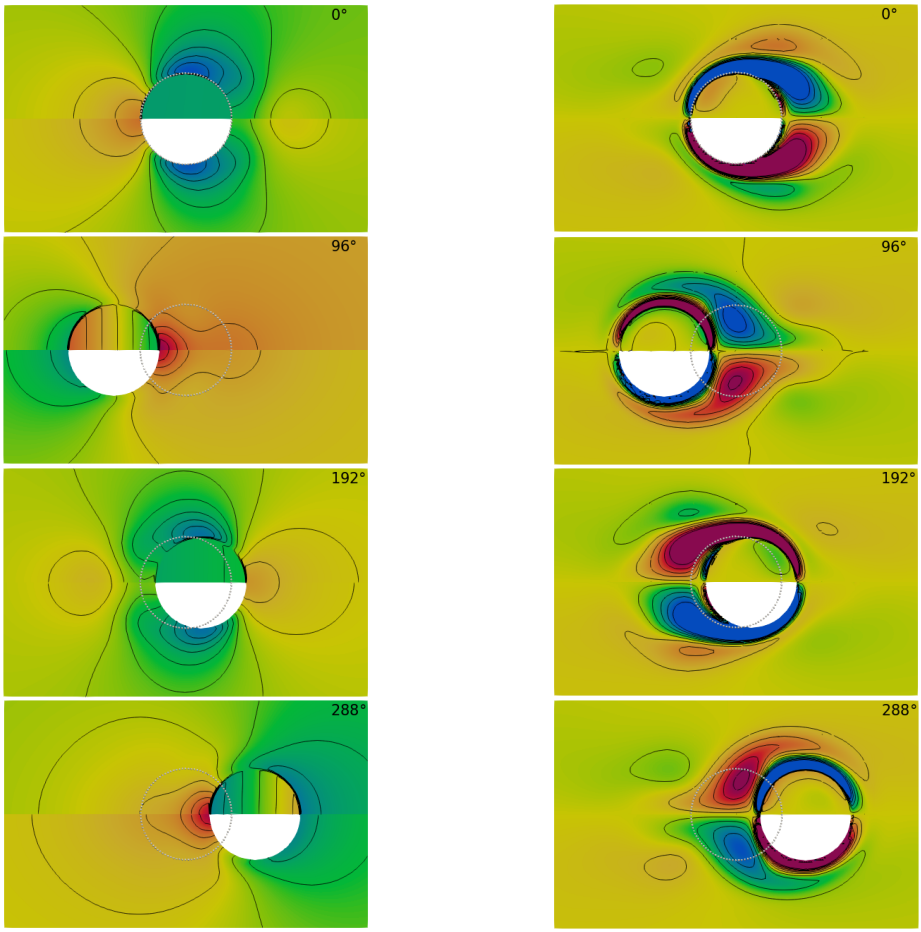

Figure 8: The pressure (left) and vorticity (right) contours at four different phase angles (0°, 96°, 192°, 288°) for the circular cylinder oscillating in quiescent flow. The top half shows the results from the IBM solver, and the bottom half shows those from the ALE solver.

Fig.9 illustrates a comparison between ALE (dashed line), IBM (solid line), and experimental [46] results for the u-component (left) and v-component (right). Four x positions and two different phase angles are shown. The u and v components are expected to exhibit symmetric and anti-symmetric velocity distributions, respectively, along the y-axis. However, due to measurement uncertainty, the velocity distribution in the experimental results along the y-axis is slightly non-symmetric. Despite this, the results generally exhibit good agreement.

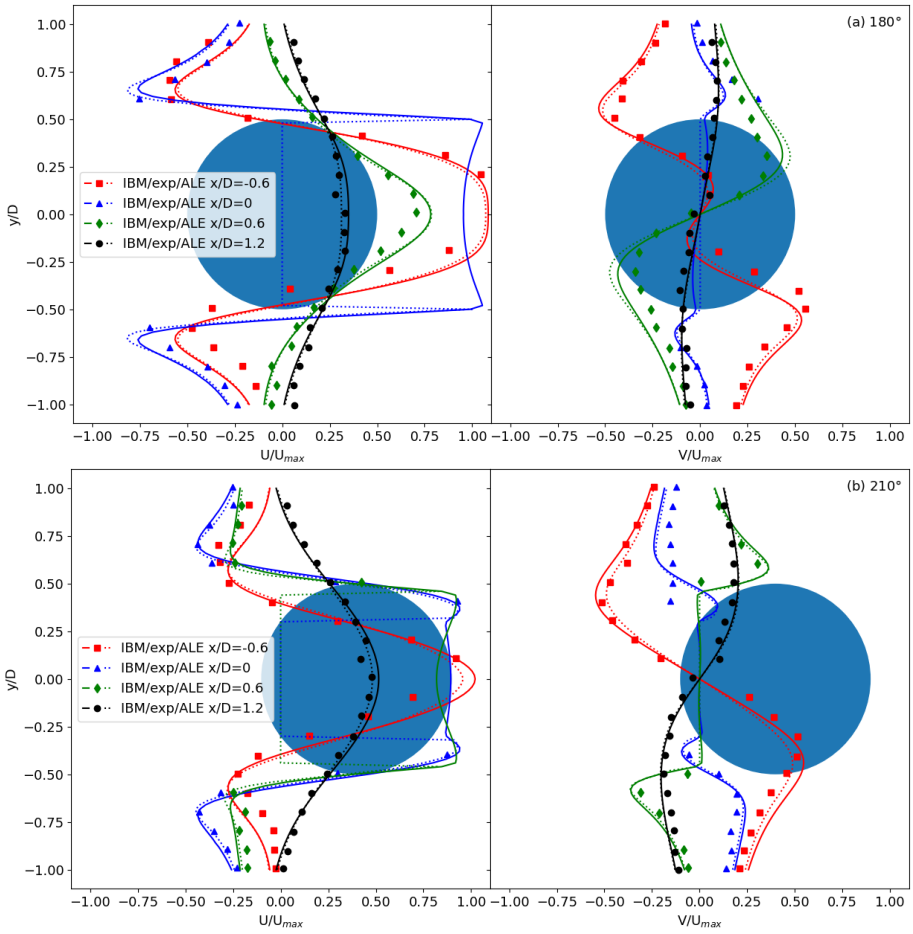

Figure 9: Comparison of velocity profiles for flow around a circular cylinder oscillating in quiescent flow obtained from ALE (dashed line), IBM (solid line), and experiments (squares) [46]. Four cross-sections x/D=-0.6, 0, 0.6, and 1.2 and two phase angles (a) 180^o (b) 210^o are shown.

Accurate predictions of the drag force are crucial in the analysis of bodies moving within a fluid. Fig. 10 shows the temporal variation of the drag coefficient obtained from experiments (black solid circles), IBM (blue solid line), and ALE (orange dashed line). The figure shows that the IBM prediction for the drag coefficient is very close to the ALE results with a maximum relative error over the considered simulation time of less than 7%. The more commonly reported error at peak C_d is just below 1.2%. Furthermore, both solver results are in excellent agreement with the experimental results [46].

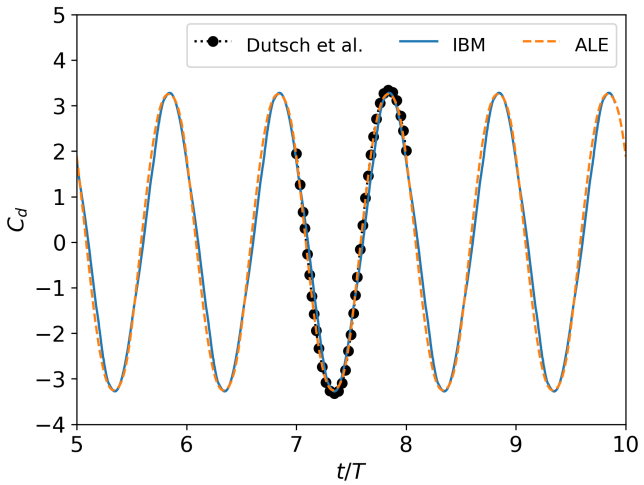

Figure 10: Convergence history of drag coefficient for flow around a circular cylinder oscillating in quiescent flow

5.3. Flow around a circular cylinder oscillating transversely

The motion of the cylinder center along the vertical direction is given by

$$y(t) = -Asin(2\pi ft), \tag{59}$$

which is perpendicular to the uniform inflow velocity U. The computational domain extends $-20D \leq x \leq 80D$ and $-40D \leq x \leq 40D$ in the streamwise and transverse directions, respectively, with the cylinder located at the origin. A uniform mesh is used for the IBM solver in the region $-2D \leq x \leq 6D$ and $-2D \leq x \leq 2D$ with 160×400 cells (spacing $D/50$) and the other

regions are discretized with a gradually increasing mesh spacing for a total of 352,000 cells. The conforming mesh for the ALE solver is again created using the *snappyHexmesh* tool as described in section 5.1.1. In the simulation of IBM, 320 Lagrangian markers are uniformly distributed in the circumferential direction around the cylinder; the corresponding ratio between the fluid grid spacing and the Lagrangian marker distance is approximately 2..

At the inflow, a uniform velocity is prescribed. On the lateral boundaries, we use a slip wall BC and for the outlet, we use a Neumann BC for velocity and a Dirichlet BC for pressure.

The Reynolds number of the flow is $Re = UD/\nu = 185$ and we consider a total of sixteen different cases: four different normalized oscillating amplitude values $A/D = 0.2$, 0.3, 0.4, and 0.5, and four different frequency ratios $f_r = 0.8$, 1, 1.1, and 1.2 with the frequency ratio defined as $f_r = f/f_0$ where f_0 is the vortex shedding frequency of the static cylinder.

All cases are run with a nominal CFL number of 0.25 with an observed maximum CFL of around 0.8 for the case with maximum amplitude and frequency.

Fig. 11 compares the primary and secondary shedding frequencies obtained by the FFT on the lift coefficient at different frequency ratios between IBM and ALE. For different oscillating frequency ratios and amplitudes, IBM captures the first and second vortex shedding accurately. The results also agree well with those of [47] and [48], which are not shown here. In addition, the drag coefficients of IBM and ALE agree well as shown in table 2, especially when the oscillating frequency is low. The maximum error of the drag coefficient is less than 4%, and the average error of all the cases is 2.2%.

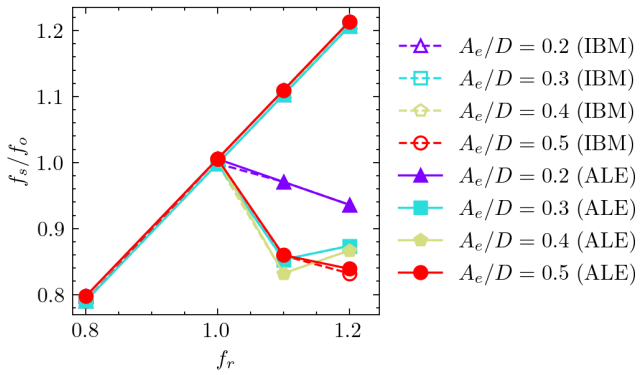

Figure 11: Comparison of the normalized frequencies for flow past around a circular cylinder oscillating transversely

Table 2: Drag coefficients of different frequency ratio for flow past around a circular cylinder oscillating transversely

A_e/D	0.2		0.3		0.4		0.5	
f/f_0	ALE	IBM	ALE	IBM	ALE	IBM	ALE	IBM
0.8	1.251	1.277	1.361	1.377	1.469	1.476	1.575	1.573
1	1.574	1.592	1.715	1.714	1.846	1.839	1.985	1.965
1.1	1.397	1.449	1.375	1.420	1.554	1.595	1.710	1.885
1.2	1.391	1.442	1.466	1.506	1.641	1.706	1.782	1.875

6. Analysis of several IBM aspects and cost reduction

The IBM validation using the ALE solver with essentially identical numerical parameters as presented in the preceding section enables us to analyze the influence of the force correction method and the Rhie-Chow interpolation on the IBM solver accuracy. We also investigate how the computational cost of the iterative IBM solver can be reduced without losing too much accuracy by simply limiting the number of outer iterations.

6.1. Force correction

The force correction was introduced in section 3.3 as a means to reduce the slip error arising from the non-reciprocal relationship between the discrete interpolation and distribution operators. Here, we use the ALE benchmark results to demonstrate that force correction is indeed important to obtain accurate IBM results. We first consider the confined stationary cylinder case used in section 4.2. The mesh for the ALE solver is generated by extracting the cylinder from the IBM mesh (see Fig. 4) using the snappyHexmesh tool. The resulting grid is shown in Fig. 12.

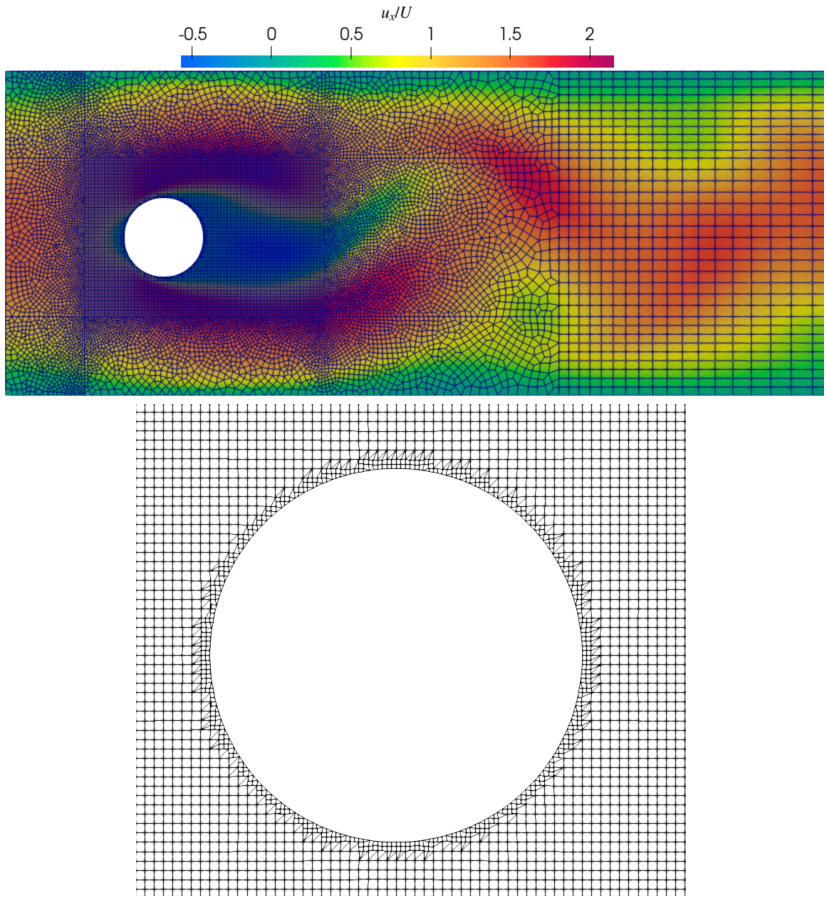

Figure 12: Inflow region of the unstructured grid (top) for the confined cylinder cases used by the ALE solver and a close-up of the cylinder region (bottom).

We have compared the ALE results with the IBM results using the iterative force correction approach proposed by Ji et al. [19], the simpler approach of Gsell et al. [12, 32], and a no-correction case. Table 3 shows a summary of the drag and lift coefficient results and it is clear that the force correction methods improve the results significantly. Gsell's method produces results that are almost identical to those obtained from the iterative method if more than about 10 iterations are used. Thus, Gsell's correction method is preferable since it is computationally much more efficient than the iterative approach while delivering the same results.

Table 3: Comparison of the drag and lift coefficients between the iterative and the correction factor method

Method	C_d	C_l
ALE	3.22547	0.98993
Ji 100	3.31391	0.96646
Ji 50	3.31356	0.96786
Ji 20	3.31357	0.96870
Ji 10	3.31334	0.96904
Ji 5	diverged	diverged
Gsell's method	3.31228	0.97089
no correction	3.60064	0.82967

We have found that force correction is even more critical for unsteady problems such as the oscillating cylinder in quiescent flow case from section 5.2. Fig. 13 shows a comparison of the temporal variation of the drag coefficient including results from IBM with force correction (using Gsell's [12] method) and IBM without force correction. Without force correction, the magnitude of the drag coefficient is over-predicted and a significant phase shift can be observed.

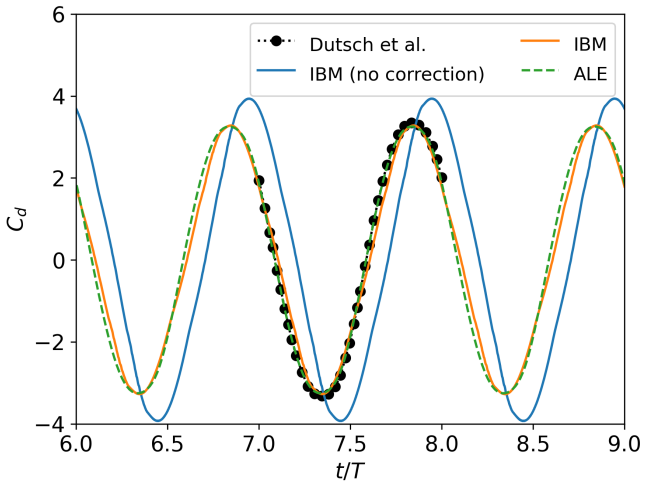

Figure 13: Comparison of the drag history predicted with and without force correction in the cylinder oscillating in quiescent surroundings test case.

6.2. Rhie-Chow interpolation

The RC interpolation is a crucial aspect of the finite volume method for collocated grids [29, 34]. IBM solvers for collocated grids must also account for RC of the body force as discussed in section 3.4. To demonstrate the importance of RC interpolation for the body force term, we consider again the confined stationary cylinder case as was used in sections 4.2 and 6.1.

For simulations using a non-dimensional time-step $\Delta t/T = 1.5\times10^{-4}$ we find that the relative error of the drag coefficient without body force RC interpolation is 15.8% whereas with RC interpolation it is only 2.3%. Moreover, we have observed that without body force RC the error is independent of the time-step but with RC interpolation we see a clear convergence with decreasing time-step towards the ALE result (not shown).

6.3. Analysis of optimum number of iterations

The implicit IBM solver proposed in this study can significantly increase the CFL number compared to explicit methods, but larger CFL number simulations require more outer iterations to meet the velocity convergence criterion within each time step. An obvious question is to ask how many outer iterations (i.e.,

how tight of a convergence criterion) are required to achieve a given accuracy when compared to ALE results. To answer this question, we consider again the confined stationary cylinder case from sections 4.2 and 6.1, and run it with a time step of $\Delta t/T = 1.5 \times 10^{-4}$ which corresponds to $CFL_{max} \approx 1.6$.

Table 4 shows the drag and lift coefficients with different outer iteration numbers for the confined stationary cylinder case. The results show that using only one outer iteration causes significantly larger errors than using multiple outer iterations. This is because, with one outer iteration, the IBM solver reverts to first-order temporal accuracy since the non-linear convective term is only approximated to first-order $\left[\dot{V}\mathbf{u}\right]_f^{n+1} \approx \dot{V}_f^n \mathbf{u}_f^{n+1}$. For this unsteady flow over a stationary cylinder case, increasing the number of outer iterations beyond two hardly improves the IBM solver accuracy.

Table 4: Drag and lift coefficients with different outer iteration numbers ($\mathrm{CFL}_{max} = 1.6$)

Method	$\mathrm{C_d}$	E_{C_d} [%]	$\mathrm{C_l}$	E_{C_l}
ALE	3.22547	NA	0.98993	NA
IBM 30	3.31228	2.69	0.97089	1.92
IBM 5	3.31228	2.69	0.97089	1.92
IBM 2	3.31228	2.69	0.97083	1.93
IBM 1	3.32728	3.16	1.03808	4.86

7. Conclusions

The validation of IBM solvers often encounters a common challenge. When validated against experimental data, uncertainties stemming from 3D effects and boundary conditions introduce complexities in replicating exact physical conditions within simulations. Alternatively, benchmark data is frequently sourced from diverse research groups utilizing different numerical techniques, grid resolutions, domain sizes, and boundary conditions. In many cases, an IBM solver is deemed "validated" if its outcomes roughly align with a limited subset of reference results, a tolerance that often prevents in-depth exploration of specific

aspects of the IBM methodology. Consequently, a consistent and more rigorous benchmarking approach for IBM solvers is needed.

Motivated by this, we introduced a novel approach to benchmark IBM solvers. Our strategy involves using a body-conforming ALE solver to validate the IBM solver, by employing the same underlying numerical framework and using identical simulation parameters to the extent possible.

We implemented a temporally second-order ALE solver based on an iterative PISO algorithm and verified the second-order temporal convergence rate. Subsequently, we devised a new iterative IBM solver built upon the ALE solver framework. We demonstrated that the new IBM solver maintained the second-order temporal convergence of the underlying ALE solver.

We validated the proposed IBM solver against the ALE solver by keeping all relevant settings identical except for the grids which differ only in the small region surrounding the immersed boundary. Our rigorous validation process encompassed static and moving immersed boundaries across a range of Reynolds numbers. The IBM solver demonstrated remarkable accuracy and robustness across steady-state and transient flows, effectively spanning a wide spectrum from low to high Reynolds numbers. For the $Re = 40$ stationary cylinder with a steady-state wake, the relative error in the drag coefficient was 0.47%. For the higher Reynolds number cases, $Re = 100$ and $Re = 185$, with unsteady vortex shedding the error increased with the Reynolds number but remained below 3.3%. A total of 17 cases with oscillating cylinders have been studied and the relative error of the drag coefficient remained below 7% with most cases having errors below 4%.

The consistent ALE benchmark results allowed us to analyze several aspects of the new IBM solver. We investigated the influence of the slip error arising from the non-reciprocity of interpolation and distribution operations commonly observed in diffusive IBM. Without mitigation of the slip error (i.e. without force correction), the quality of the prediction for the drag coefficient severely deteriorated with an error of almost 12%. With proper force correction, the error is reduced to below 2.7%. We could also show that the simpler force correction

method of Gsell et al. [12] delivers the same accuracy as the computationally more expensive iterative method of Ji et al. [19] for the considered cases.

In the context of collocated FVM, proper Rhie-Chow interpolation of the body force in IBM was of paramount importance. This is due to the additional body force's large magnitude and its rapid temporal and spatial variation. We employed a reconstruction technique for pressure gradient and body force terms in the momentum equation for the Rhie-Chow interpolation approach and showed that without it the IBM solver reduced to zeroth-order temporal convergence with a much larger error.

Our exploration of the iterative IBM solver unveiled its capability to enhance the CFL limit due to its implicit attributes. This allows the user to reduce the overall computational cost of unsteady IBM simulations by finding an optimum between the size of the time step (i.e. the CFL number) and the number of outer iterations. For the considered test case of flow over a confined stationary cylinder with $Re = 100$, we found that with $CFL_{max} \approx 1.6$, two outer iterations are sufficient, and more iterations do not further improve accuracy.

The IBM solver developed in this work was built on a conforming mesh ALE solver. In theory, one could thus consider a hybrid IBM-ALE solver where some boundaries are treated with a conforming mesh (with or without mesh motion) and some boundaries are treated by IBM. This could be beneficial for problems where some boundaries have much larger displacement than others. For example, in simulations of the left ventricle, the motion of the ventricle wall is small enough to be treated with ALE whereas the large motion of the heart valves could be described by IBM. A detailed analysis of such a hybrid ALE-IBM solver will be presented in a future publication.

Acknowledgments

Lianxia Li and Michael Stoellinger are partially funded by the University of Wyoming's "School of Energy Resources" through the "Wind Energy Research Center". Lianxia Li is partially funded by the Wyoming INBRE Graduate

Research Assistantships.

References

[1] C. S. Peskin, Flow patterns around heart valves: a numerical method, Journal of computational physics 10 (1972) 252–271.

[2] E. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, Journal of computational physics 161 (2000) 35–60.

[3] M. Uhlmann, An immersed boundary method with direct forcing for the simulation of particulate flows, Journal of computational physics 209 (2005) 448–476.

[4] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (2005) 239–261.

[5] F. Sotiropoulos, X. Yang, Immersed boundary methods for simulating fluid–structure interaction, Progress in Aerospace Sciences 65 (2014) 1–21.

[6] W. X. Huang, F. B. Tian, Recent trends and progress in the immersed boundary method, Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science 233 (2019) 7617–7636.

[7] W. Kim, H. Choi, Immersed boundary methods for fluid-structure interaction: A review, International Journal of Heat and Fluid Flow 75 (2019) 301–309.

[8] R. Verzicco, Immersed boundary methods: Historical perspective and future outlook, Annual Review of Fluid Mechanics 55 (2023).

[9] S. Murakami, A. Mochida, On turbulent vortex shedding flow past 2d square cylinder predicted by cfd, Journal of Wind Engineering and Industrial Aerodynamics 54 (1995) 191–211.

[10] L. Bruno, D. Fransos, N. Coste, A. Bosco, 3d flow around a rectangular cylinder: a computational study, Journal of Wind Engineering and Industrial Aerodynamics 98 (2010) 263–276.

[11] T. Seta, K. Hayashi, A. Tomiyama, Analytical and numerical studies of the boundary slip in the immersed boundary-thermal lattice boltzmann method, International Journal for Numerical Methods in Fluids 86 (2018) 454–490.

[12] S. Gsell, U. d'Ortona, J. Favier, Explicit and viscosity-independent immersed-boundary scheme for the lattice boltzmann method, Physical Review E 100 (2019) 033306.

[13] J. Kettemann, I. Gatin, C. Bonten, Verification and validation of a finite volume immersed boundary method for the simulation of static and moving geometries, Journal of Non-Newtonian Fluid Mechanics 290 (2021) 104510.

[14] X. Zhang, X. Gu, N. Ma, A ghost-cell immersed boundary method on preventing spurious oscillations for incompressible flows with a momentum interpolation method, Computers & Fluids 220 (2021) 104871.

[15] K. Kingora, H. Sadat-Hosseini, A novel interpolation-free sharp-interface immersed boundary method, Journal of Computational Physics 453 (2022) 110933.

[16] D. Z. Noor, M.-J. Chern, T.-L. Horng, An immersed boundary method to solve fluid–solid interaction problems, Computational Mechanics 44 (2009) 447–453.

[17] A. Pinelli, I. Naqavi, U. Piomelli, J. Favier, Immersed-boundary methods for general finite-difference and finite-volume navier–stokes solvers, Journal of Computational Physics 229 (2010) 9073–9091.

[18] R.-Y. Li, C.-M. Xie, W.-X. Huang, C.-X. Xu, An efficient immersed boundary projection method for flow over complex/moving boundaries, Computers & Fluids 140 (2016) 122–135.

[19] C. Ji, A. Munjiza, J. Williams, A novel iterative direct-forcing immersed boundary method and its finite volume applications, Journal of Computational Physics 231 (2012) 1797–1821.

[20] L. Nicolaou, S. Jung, T. Zaki, A robust direct-forcing immersed boundary method with enhanced stability for moving body problems in curvilinear coordinates, Computers & Fluids 119 (2015) 101–114.

[21] Ž. Tuković, M. Perić, H. Jasak, Consistent second-order time-accurate non-iterative piso-algorithm, Computers & Fluids 166 (2018) 78–85.

[22] A. Vreman, Immersed boundary and overset grid methods assessed for stokes flow due to an oscillating sphere, Journal of Computational Physics 423 (2020) 109783.

[23] K. Schäfer, P. Forooghi, S. Straub, B. Frohnapfel, A. Stroh, Direct numerical simulations of a turbulent flow over wall-mounted obstacles—a comparison of different numerical approaches, in: Direct and Large Eddy Simulation XII, Springer, 2020, pp. 91–96.

[24] F. Capuano, N. Beratlis, F. Zhang, Y. Peet, K. Squires, E. Balaras, Cost vs accuracy: Dns of turbulent flow over a sphere using structured immersed-boundary, unstructured finite-volume, and spectral-element methods, European Journal of Mechanics-B/Fluids 102 (2023) 91–102.

[25] Ž. Tuković, H. Jasak, A moving mesh finite volume interface tracking method for surface tension dominated interfacial fluid flow, Computers & fluids 55 (2012) 70–84.

[26] T. Gillebaart, D. Blom, A. van Zuijlen, H. Bijl, Time consistent fluid structure interaction on collocated grids for incompressible flow, Computer Methods in Applied Mechanics and Engineering 298 (2016) 159–182.

[27] C. M. Rhie, W.-L. Chow, Numerical study of the turbulent flow past an airfoil with trailing edge separation, AIAA journal 21 (1983) 1525–1532.

[28] A. Pascau, Cell face velocity alternatives in a structured colocated grid for the unsteady navier–stokes equations, International Journal for Numerical Methods in Fluids 65 (2011) 812–833.

[29] M. Darwish, F. Moukalled, The finite volume method in computational fluid dynamics: an advanced introduction with OpenFOAM® and Matlab®, Springer, 2016.

[30] J. H. Ferziger, M. Peric, Computational methods for fluid dynamics. 3. rev, Springer, 2002.

[31] C. J. Greenshields, H. G. Weller, Note on Computational Fluid Dynamics: General Principles, CFD Direct https://cfd. direct., 2022.

[32] S. Gsell, J. Favier, Direct-forcing immersed-boundary method: a simple correction preventing boundary slip error, Journal of Computational Physics 435 (2021) 110265.

[33] A. M. Roma, C. S. Peskin, M. J. Berger, An adaptive version of the immersed boundary method, Journal of Computational Physics 153 (1999) 509–534.

[34] S. Zhang, X. Zhao, S. Bayyuk, Generalized formulations for the rhie–chow interpolation, Journal of Computational Physics 258 (2014) 880–914.

[35] H. Weller, A. Shahrokhi, Curl-free pressure gradients over orography in a solution of the fully compressible euler equations with implicit treatment of acoustic and gravity waves, Monthly Weather Review 142 (2014) 4439–4457.

[36] H. J. Aguerre, C. I. Pairetti, C. M. Venier, S. M. Damián, N. M. Nigro, An oscillation-free flow solver based on flux reconstruction, Journal of Computational Physics 365 (2018) 135–148.

[37] A. Moraes, P. Lage, G. Cunha, L. da Silva, Analysis of the non-orthogonality correction of finite volume discretization on unstructured meshes, in: Proceedings of the 22nd International Congress of Mechanical Engineering, volume 3, 2013, pp. 3519–3530.

[38] H. Jasak, Error analysis and estimation for the finite volume method with applications to fluid flows, Ph.D. thesis, Imperial College London, 1996.

[39] L. Shen, E.-S. Chan, P. Lin, Calculation of hydrodynamic forces acting on a submerged moving object using immersed boundary method, Computers & Fluids 38 (2009) 691–702.

[40] A. P. S. Bhalla, R. Bale, B. E. Griffith, N. A. Patankar, A unified mathematical framework and an adaptive numerical method for fluid–structure interaction with rigid, deforming, and elastic bodies, Journal of Computational Physics 250 (2013) 446–476.

[41] R. Gautier, D. Biau, E. Lamballais, A reference solution of the flow over a circular cylinder at re= 40, Computers & Fluids 75 (2013) 103–111.

[42] D. S. Lee, M. Y. Ha, S. J. Kim, H. S. Yoon, Application of immersed boundary method for flow over stationary and oscillating cylinders, Journal of mechanical science and technology 20 (2006) 849–863.

[43] C.-C. Liao, Y.-W. Chang, C.-A. Lin, J. McDonough, Simulating flows with moving rigid boundary using immersed-boundary method, Computers & Fluids 39 (2010) 152–167.

[44] J. Yang, F. Stern, A simple and efficient direct forcing immersed boundary framework for fluid–structure interactions, Journal of Computational Physics 231 (2012) 5029–5061.

[45] J. Qin, E. M. Kolahdouz, B. E. Griffith, An immersed interface-lattice boltzmann method for fluid-structure interaction, Journal of Computational Physics 428 (2021) 109807.

[46] H. DÜTSCH, F. DURST, S. BECKER, H. LIENHART, Low-reynolds-number flow around an oscillating circular cylinder at low keulegan–carpenter numbers, Journal of Fluid Mechanics 360 (1998) 249–271.

[47] P. Anh-Hung, L. Chang-Yeol, S. Jang-Hoon, C. Ho-Hwan, K. Hee-Jung, H.-S. Yoon, D.-W. Park, I.-R. Park, Laminar flow past an oscillating circular cylinder in cross flow, Journal of Marine Science and Technology 18 (2010) 5.

[48] I. Cheylan, J. Favier, P. Sagaut, Immersed boundary conditions for moving objects in turbulent flows with the lattice-boltzmann method, Physics of Fluids 33 (2021) 095101.