dynamics

MDPI

Article

Implementation and Validation of Explicit Immersed Boundary Method and Lattice Boltzmann Flux Solver in OpenFOAM

Yangyang Liu [1], Ziying Zhang [2], Hua Zhang [1] and Yaguang Liu [1,*]

[1] Department of Mechanical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260, Singapore
[2] College of Mechanical Engineering, Chongqing University of Technology, Chongqing 400054, China
* Correspondence: yaguang.liu@u.nus.edu

Abstract: In this work, the explicit boundary-condition-enforced immersed boundary method (EIBM) and the lattice Boltzmann flux solver (LBFS) are integrated into OpenFOAM to efficiently solve incompressible flows with complex geometries and moving boundaries. The EIBM applies the explicit technique to greatly improve the computational efficiency of the original boundary-condition-enforced immersed boundary method. In addition, the improved EIBM inherits the accurate interpretation of the no-slip boundary condition and the simple implementation from the original one. The LBFS uses the finite volume method to discretize the recovered macroscopic governing equations from the lattice Boltzmann equation. It enjoys the explicit relationship between the pressure and density, which avoids solving the pressure Poisson equation and thus saves much computational cost. Another attractive feature of the LBFS lies in its simultaneous evaluation of the inviscid and viscous fluxes. OpenFOAM, as an open-source CFD platform, has drawn increasing attention from the CFD community and has been proven to be a powerful tool for various problems. Thus, implementing the EIBM and LBFS into such a popular platform can advance the practical application of these two methods and may provide an effective alternative for complicated incompressible flow problems. The performance of the integrated solver in OpenFOAM is comprehensively assessed by comparing it with the widely used numerical solver in OpenFOAM, namely, the Pressure-Implicit with Splitting of Operators (PISO) algorithm with the IBM. A series of representative test cases with stationary and moving boundaries are simulated. Numerical results confirm that the present method does not have any streamline penetration and achieves the second-order accuracy in space. Therefore, the present method implemented in the open-source platform OpenFOAM may have good potential and can serve as a powerful tool for practical engineering problems.

Keywords: immersed boundary method; OpenFOAM; moving boundary; lattice Boltzmann flux solver

Citation: Liu, Y.; Zhang, Z.; Zhang, H.; Liu, Y. Implementation and Validation of Explicit Immersed Boundary Method and Lattice Boltzmann Flux Solver in OpenFOAM. *Dynamics* **2024**, *4*, 14–39. https://doi.org/10.3390/dynamics4010002

Academic Editor: Christos Volos

Received: 14 November 2023
Revised: 6 December 2023
Accepted: 21 December 2023
Published: 3 January 2024

1. Introduction

Industries and natural phenomena involve many flow problems with moving boundaries and fluid–structure interactions (FSIs) [1–6]. Accurate modelling of such complex problems enjoys of great importance for investigations of the underlying mechanisms. Generally, two types of approaches, namely, the boundary-conforming approach and boundary-non-conforming approach, are applied by the computational fluid dynamics (CFD) community [7–11]. One popular boundary-non-conforming method is the immersed boundary method (IBM), which was first proposed by Peskin [12]. Instead of using body-fitted meshing methods to conform to the boundaries or interfaces as with the boundary-conforming approach, IBM adopts the fixed Eulerian mesh for resolving the flow field and the discrete Lagrangian points to describe the immersed boundaries. Conditions on boundaries or interfaces are enforced via the restoring forces exerted on the discrete Lagrangian points by the boundaries. These forces further correct the flow variables resolved on the fixed Eulerian mesh as external body forces. In this way, the boundary

treatment is decoupled with the mesh generation in IBM, providing great flexibility and ease in simulating flow problems with complicated boundaries. In addition, avoiding mesh movement and mesh re-generation for problems with moving boundaries makes IBM more efficient than the boundary-conforming approach.

Due to the simplicity and flexibility of the IBM, various variants have been proposed since the original IBM. In fact, the original IBM [12] is categorized as the penalty-forcing scheme that calculates the restoring force on the Lagrangian points through Hooke's law [13–15] with a user-defined stiffness constant. Another category is the feedback-forcing scheme [16], in which the restoring force is evaluated using control theory. Following these two types of IBM approaches, the direct forcing scheme [17–19] and momentum exchange scheme [20] were developed. These two IBM versions eliminate the user-defined parameter and improve the physical robustness. In this regard, the direct forcing scheme has been integrated into the open-source CFD toolbox, OpenFOAM [21,22], and applied to solve many FSI problems.

In the original direct forcing method developed in Refs. [23,24], to satisfy the no-slip boundary condition, the restoring force term is evaluated by setting the velocity at the immersed boundary point to the desired velocity in the discrete momentum equation to avoid feedback adjustments. It is a local solution reconstruction process, and an explicit forcing term is not required in the momentum equations at all. Following the original principle of the direct forcing scheme, by incorporating the idea of Peskin's method, another type of direct forcing scheme with the Dirac delta function is developed [19,25]. The main idea is briefly introduced as follows. Firstly, the velocities at the Lagrangian points are interpolated from the surrounding Eulerian points. Then, the local force is determined by imposing the no-slip boundary condition at the Lagrangian points. Lastly, these forces are distributed to the Euler points and substitute them into the momentum equation. Generally, the types of direct forcing methods calculate the force density at the boundary points via the momentum equation explicitly. This is the key difference between the direct forcing method and other IBMs, such as the penalty method [13–15] and the velocity-correction-based IBM [10,11], which will be discussed later. Nowadays, the direct forcing method has been incorporated into various numerical frameworks, such as the projection method [26] and the lattice Boltzmann method (LBM) [27] to tackle complex FSI problems. Nevertheless, unphysical streamline penetration through the immersed interface or boundary may appear, which means that the no-slip boundary condition is not exactly satisfied. This may be caused by the explicit evaluation of the restoring force first at the beginning of each time step in some direct forcing methods. In particular, the flow field obtained from such an approximated restoring force may not guarantee the no-slip wall boundary condition.

To tackle the streamline penetration issues, the multi-direct forcing scheme [28,29] is proposed, and it applies an iterative implementation of the direct forcing IBM for accurate satisfaction of the no-slip boundary condition. However, the convergence of the iterations may affect the computational efficiency of this method. Without any iterative process, Shu and Wu [10,11] developed the boundary-condition-enforced IBM. In this method, the velocity correction is implicitly resolved based on the no-slip boundary condition, and then it is used to evaluate the restoring force through Newton's second law. As a result, the no-slip boundary condition can be exactly enforced, and no unphysical streamline penetration is produced. It is noteworthy that in practical implementation of this implicit IBM (IIBM), calculating the velocity correction involves solving a linear system of equations assembled according to the information on the Eulerian and Lagrangian meshes. The large-scale problems will result in a large matrix to be solved. When the moving boundary is considered, the matrix should be constructed and solved implicitly at each time step. In these cases, the implicit resolving strategy of IIBM may be inefficient, which limits its widespread application. Recently, Zhao et al. [30] introduced the explicit technique to improve the efficiency of the IIBM. Through the accuracy analysis, the terms with higher-than-second-order accuracy for solving the velocity correction in the IIBM process can be discarded. Then, the matrix of the linear equation system in IIBM can be simplified to a diagonal

matrix without affecting the global second-order accuracy in space. Because considering the high-order terms will cost much time, such a reasonable exclusion of the high-order terms can reduce the time complexity and improve the efficiency without preserving the spatial second-order accuracy as the flow field solver. Finally, the velocity correction is directly obtained in an explicit way. This explicit boundary-condition-enforced IBM (EIBM) has a computational complexity of $O(M)$, with M being the number of Lagrangian points on the immersed boundary, which indicates a better efficiency than the IIBM of $O(M^2)$. Its applications to solve flow problems with moving boundaries and FSI problems [30] confirm the improvement in computational efficiency.

In fact, another important issue in the simulation of incompressible flow problems with complex boundaries or moving boundaries lies in the incompressible flow solver. One popular strategy is to incorporate IBMs with the Navier–Stokes (NS) solvers [24,31]. Although there have been many successful applications of immersed boundary–NS (IB-NS) solvers, the NS solvers are bound to treat the velocity–pressure coupling in incompressible flow simulations. Normally, a pressure Poisson equation should be solved, which may degrade the computational efficiency. Another alternative enjoying increasing popularity is the incorporation of the IBM into the LBM [10,11,20,32]. Owing to the simplicity and efficiency of LBM [33], the combination of the IBM and LBM has been applied to solve various incompressible flow problems with complicated boundaries or moving boundaries [34,35]. However, it should be noted that LBM has some intrinsic disadvantages [36], such as the requirement of uniform meshes and the tie-up between the time step and lattice spacing. Limitations of uniform meshes and relatively small time steps could result in high memory consumption and poor computational performance of the immersed boundary-lattice Boltzmann method (IB-LBM), thus hindering its effective simulation of moving boundary problems or complex FSI problems.

Apart from the IB-NS solvers and IB-LBMs, Wang et al. [37,38] combine the IIBM [10] with the lattice Boltzmann flux solver [39] (IIB-LBFS) for FSI problems. As a recently developed flow solver, the LBFS inherits advantages from the NS solver and LBM, thus overcoming drawbacks of these two types of approaches [40–44]. To be specific, the LBFS is under the finite volume framework, and the macroscopic flow variables, rather than the density distribution function, are directly evolved. Inviscid and viscous fluxes at the cell interface are simultaneously evaluated by the local reconstruction of the standard LBM solutions. Such a local application of LBM provides the LBFS with the great flexibility of using non-uniform meshes as well as the independence of the time step on the lattice spacing [45–47]. Furthermore, the explicit relationship between the density and the pressure avoids solving the Poisson equation for pressure, like the NS solvers. In the work of Wang et al. [38], the performance of the IIB-LBFS has been validated and assessed by solving FSI problems. Nonetheless, as discussed above, the IIBM suffers from the tedious matrix assembly and inefficient implicit resolving strategy, while the EIBM is simpler and more efficient. Therefore, there is motivation to combine the EIBM with the LBFS and explore its capacity for simulating problems with complex boundaries or moving boundaries. Compared with the work of Zhao et al. [30] where the D1Q4 lattice Boltzmann model [48] is applied to calculate the inviscid flux and the central difference approximation is used for the viscous flux evaluation, the LBFS can evaluate the inviscid and viscous fluxes at the same time. No extra treatment of the viscous flux may save some computational efforts and make the EIB-LBFS more consistent.

In this work, the EIB-LBFS will be integrated into the open-source CFD toolbox OpenFOAM. A set of representative flow problems with complex stationary or moving boundaries will be tested to validate and examine the performance of the EIB-LBFS. The commonly used direct forcing IBM [21,22] embedded in OpenFOAM will also be used in these tests for comparison purposes. The rest of this paper is organized as follows. In Section 2, the EIB-LBFS for incompressible flows with moving boundaries is described. Section 3 is devoted to clarification of implementing the EIB-LBFS in OpenFOAM. Then,

a series of benchmark tests are conducted in Section 4 to assess the performance of the EIB-LBFS. This paper is finalized with some conclusions in Section 5.

2. Explicit Immersed Boundary-Lattice Boltzmann Flux Solver for Incompressible Flows with Moving Boundaries

The EIB-LBFS decouples the resolving process of flow field variables and the imposition of boundary effects into two fractional steps, namely, the predictor step and the corrector step. In the predictor step, the intermediate flow field is resolved by applying the LBFS without considering the boundary effects. The intermediate flow field is then corrected via employing the EIBM in the corrector step. Details of these two procedures will be clarified in this section.

2.1. Governing Equations and Finite Volume Discretization

The governing equations for incompressible flows within the low Mach number limit can be written as follows:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \tag{1}$$

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u} + p\boldsymbol{I}) = \nabla \cdot \left[\mu\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T\right)\right] + \boldsymbol{f} \tag{2}$$

where ρ, $\boldsymbol{u} = (u, v, w)$, p, and μ are the density, velocity vector, pressure, and dynamic viscosity of the fluid flow, respectively. $\boldsymbol{I}$ denotes the unit tensor. $\boldsymbol{f}$ is the restoring force on the Eulerian mesh points interpreting the boundary effects. Equations (1) and (2) can be rearranged into the following form:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{F} = \boldsymbol{G} \tag{3}$$

where $\boldsymbol{U}$, $\boldsymbol{F}$, and $\boldsymbol{G}$ represent the vector of conservative variables, the flux tensor, and the body force vector, respectively. After applying the divergence theorem, the integral form of Equation (3) over a control cell Ω_i can be written as

$$V_i \frac{d}{dt}\boldsymbol{U}_i = -\sum_{j=1}^{nf} (\boldsymbol{F}_{n,k} A)_j + \boldsymbol{G}_i \tag{4}$$

where $\boldsymbol{U}_i$, V_i, n_f, A_j, and $\boldsymbol{G}_i$ are the flow variable vector at the cell center of Ω_i, the volume of Ω_i, the number of cell interfaces in Ω_i, the j-th interface area, and the force vector at the cell center of Ω_i, respectively. $\boldsymbol{F}_n = \boldsymbol{F} \cdot \boldsymbol{n}$, and $\boldsymbol{n}$ is the outward unit vector normal to the cell interface. Note that the projection or pressure correction methods are not used here. As discussed in the Introduction, the LBFS is used to evaluate the fluxes of weakly compressible NS Equations (1) and (2), which are recovered from the lattice Boltzmann equation through the Chapman–Enskog expansion analysis. The pressure can be directly calculated from the density, which avoids solving the pressure Poisson equation. Additionally, the incompressibility can be effectively guaranteed by the limit of the small Mach number.

In this work, Equation (4) is solved through the fractional method. In the predictor step, the restoring forcing $\boldsymbol{f}$ is not considered, and the intermediate flow field is predicted using the LBFS, i.e.,

$$\boldsymbol{U}_i^* = -\frac{\Delta t}{V_i} \sum_{j=1}^{nf} (\boldsymbol{F}_{n,k} A)_j + \boldsymbol{U}_i^n \tag{5}$$

where the superscripts "*" and "n" denote the intermediate time level and the current time level, respectively. Section 2.2 will detail the flow field prediction. In the corrector step, the velocity is corrected by applying the EIBM to consider the boundary effects via

$$\rho^{n+1}\frac{\boldsymbol{u}^{n+1}-\boldsymbol{u}^{*}}{\Delta t}=\boldsymbol{f} \tag{6}$$

where the superscript "$n+1$" is the next time level. Note that ρ^{n+1} is equal to ρ^{*}, and only the fluid velocity correction $\Delta\boldsymbol{u}=\boldsymbol{u}^{n+1}-\boldsymbol{u}^{*}$ on Eulerian mesh points is to be determined according to the boundary conditions on the immersed boundaries. In practical implementation, Equation (6) is not solved directly, but the restoring force $\boldsymbol{f}$ is imposed via the velocity correction $\Delta\boldsymbol{u}$, which will be illustrated in detail in Section 2.3. Once $\Delta\boldsymbol{u}$ is available, the velocity can be updated via $\boldsymbol{u}^{n+1}=\Delta\boldsymbol{u}+\boldsymbol{u}^{*}$. Because the conventional IBMs, like the penalty-forcing methods and the direct forcing methods, calculate the restoring force $\boldsymbol{f}$ first and then $\boldsymbol{u}^{n+1}$ is obtained by solving the momentum equation with $\boldsymbol{f}^{n}$ as the source term directly, it cannot guarantee that the velocity at the boundary point interpolated from $\boldsymbol{u}^{n+1}$ fulfills the no-slip boundary condition. However, EIBM regards the force $\boldsymbol{f}$ as unknown and determines the velocity correction first by forcing the velocity at the boundary point interpolated from $\boldsymbol{u}$ to satisfy the no-slip boundary condition. Finally, the force on the boundary point is calculated by Equation (6) with the known fluid velocity correction $\Delta\boldsymbol{u}$. EIBM is an explicit form inherited from implicit boundary-condition-forcing IBM, and it can accurately satisfy the no-slip boundary condition with a low computational cost.

2.2. Flow Field Prediction through Lattice Boltzmann Flux Solver

Based on multiscale Chapman–Enskog expansion analysis, Equations (1) and (2) without the forcing term can be recovered [39] by the lattice Boltzmann equation, and the numerical fluxes at the cell interface can be expressed as follows:

$$\begin{aligned}
F_1 &= \sum_{\alpha=0}^{Nd}(\boldsymbol{e}_\alpha)_1 f_\alpha^{eq}\\
F_2 &= \sum_{\alpha=0}^{Nd}(\boldsymbol{e}_\alpha)_1(\boldsymbol{e}_\alpha)_1\left[f_\alpha^{eq}+\left(1-\tfrac{1}{2\tau_\nu}\right)f_\alpha^{neq}\right]\\
F_3 &= \sum_{\alpha=0}^{Nd}(\boldsymbol{e}_\alpha)_1(\boldsymbol{e}_\alpha)_2\left[f_\alpha^{eq}+\left(1-\tfrac{1}{2\tau_\nu}\right)f_\alpha^{neq}\right]
\end{aligned} \tag{7}$$

where $\boldsymbol{e}_\alpha$ and τ_ν denote the lattice velocity vector and single relaxation parameter, respectively. The subscripts "1" and "2" are the outward normal direction and tangential direction of the cell interface used in the local coordinate system, respectively. N_d+1 is the number of discrete particle velocities in the lattice velocity model and $N_d+1=9$ here because the D2Q9 lattice velocity model is employed for the two-dimensional cases. τ_ν in Equation (7) is obtained from the following relationship:

$$\nu=(\tau_\nu-1/2)c_s^2\delta_t \tag{8}$$

where ν is the kinematic viscosity, δ_t denotes the streaming time step, and c_s represents the sound speed. f_α^{eq} is the equilibrium density distribution function along the α direction and f_α^{neq} denotes the non-equilibrium one. f_α^{eq} can be computed based on the weights of the D2Q9 model and the flow variables (density and velocity). Its expression can refer to Ref. [39]. f_α^{neq} can be approximated based on the equilibrium density distribution functions at $\boldsymbol{r}$ and its surrounding nodes $\boldsymbol{r}-\boldsymbol{e}_\alpha\delta_t$ as

$$f_\alpha^{neq}(\boldsymbol{r},t)=-\tau_\nu\left[f_\alpha^{eq}(\boldsymbol{r},t)-f_\alpha^{eq}(\boldsymbol{r}-\boldsymbol{e}_\alpha\delta_t,t-\delta_t)\right]+O\left(\delta_t^2\right) \tag{9}$$

where $\boldsymbol{r}$ represents the physical location of the cell interface and t denotes the time.

To calculate $f_\alpha^{eq}(\boldsymbol{r}-\boldsymbol{e}_\alpha\delta_t, t-\delta_t)$, the flow variables at the location $\boldsymbol{r}-\boldsymbol{e}_\alpha\delta_t$ are interpolated from those at cell centers $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ by

$$\boldsymbol{U}(\boldsymbol{r}-\boldsymbol{e}_\alpha\delta_t) = \begin{cases} \boldsymbol{U}(\boldsymbol{r}_i) + \nabla\boldsymbol{U}(\boldsymbol{r}_i)\Delta\boldsymbol{r}_i, & (\boldsymbol{r}-\boldsymbol{e}_\alpha\delta_t) \in \Omega_i \\ \boldsymbol{U}(\boldsymbol{r}_j) + \nabla\boldsymbol{U}(\boldsymbol{r}_j)\Delta\boldsymbol{r}_j, & (\boldsymbol{r}-\boldsymbol{e}_\alpha\delta_t) \in \Omega_j \end{cases} \tag{10}$$

with

$$\Delta\boldsymbol{r}_k = \left[(\boldsymbol{r}-\boldsymbol{e}_\alpha\delta_t-\boldsymbol{r}_k)_x, (\boldsymbol{r}-\boldsymbol{e}_\alpha\delta_t-\boldsymbol{r}_k)_y\right]^T, \; k = i \, or \, j \tag{11}$$

where $\nabla\boldsymbol{U}$ denotes the gradient of the solution vector. $f_\alpha^{eq}(\boldsymbol{r}-\boldsymbol{e}_\alpha\delta_t, t-\delta_t)$ can then be computed based on the definition of the equilibrium distribution function. Furthermore, ρ and $\boldsymbol{u}$ at the location $\boldsymbol{r}$ and time t are calculated through $f_\alpha^{eq}(\boldsymbol{r}-\boldsymbol{e}_\alpha\delta_t, t-\delta_t)$ [39]. Thus, $f_\alpha^{eq}(\boldsymbol{r}, t)$ can be obtained. After computing $f_\alpha^{neq}(\boldsymbol{r}, t)$ through Equation (9), the fluxes $\boldsymbol{F} = \left(F_1, F_2, F_3\right)^T$ can finally be evaluated using Equation (7).

It is noted that the flux $\boldsymbol{F}$ is computed in the local coordinate system by the LBFS. The following transformation is applied to calculate the flux $\boldsymbol{F}_n$ in the global coordinate in Equation (4):

$$\boldsymbol{F}_n = \left(F_1, F_2n_{1x} + F_3n_{2x}, F_2n_{1y} + F_3n_{2y}, F_2n_{1z} + F_3n_{2z}\right)^T \tag{12}$$

where $\boldsymbol{n}_\beta = (n_{\beta x}, n_{\beta y})$ with β = 1 and 2 denote the unit vectors in directions 1 and 2 of the local coordinate system at the cell interface, respectively. In this way, the inviscid and viscous fluxes are simultaneously evaluated. The intermediate flow field is then predicted by solving Equation (5) with a proper time-marching strategy, like the explicit Euler scheme or the multi-stage Runge–Kutta scheme. From Equations (9) and (10), it is clear that the LBFS is second-order accurate in space, which has been proven in Refs. [39–41,43–45].

2.3. Velocity Correction through Explicit Boundary-Condition-Enforced Immersed Boundary Method

As shown in Equation (6), the velocity correction $\Delta\boldsymbol{u}$ is related to the restoring force $\boldsymbol{f}$, i.e.,

$$\rho^{n+1}\Delta\boldsymbol{u} = \Delta t\boldsymbol{f} \tag{13}$$

If $\boldsymbol{f}$ is approximated first, $\Delta\boldsymbol{u}$ can be explicitly computed, and thus the flow field $\boldsymbol{u}^{n+1}$ can be obtained. However, as discussed in the Introduction, it cannot ensure that the velocity at the boundary wall interpolated from $\boldsymbol{u}^{n+1}$ exactly fulfills the no-slip boundary condition. As a result, the streamlines may penetrate the solid body surface unphysically. To avoid this problem, Wu et al. [10] proposed an implicit velocity correction technique to accurately enforce the no-slip boundary condition.

Unlike the explicit calculation, $\boldsymbol{f}$ in Equation (13) is regarded as unknown. The velocity correction $\Delta\boldsymbol{u}$ at the Eulerian mesh is computed first. Specifically, $\Delta\boldsymbol{u}(\boldsymbol{x}_i)$ at the Eulerian control cell Ω_i can be interpolated from the velocity corrections $\Delta\boldsymbol{u}_B(\boldsymbol{X}_l)$ at the Lagrangian points $\boldsymbol{X}_l$ on the immersed boundary as follows:

$$\Delta\boldsymbol{u}(\boldsymbol{x}_i) = \sum_{l=1}^{M} D(\boldsymbol{x}_i - \boldsymbol{X}_l)\Delta\boldsymbol{u}_B(\boldsymbol{X}_l)\Delta s_l \tag{14}$$

where M denotes the number of Lagrangian points representing the boundary of solid body and Δs_l is the spatial interval between two adjacent Lagrangian points. $D(\boldsymbol{x}_i - \boldsymbol{X}_l)$ is a discrete delta function [7] given as

$$\begin{aligned} D(\boldsymbol{x}_i - \boldsymbol{X}_l) &= \tfrac{1}{h^2}\delta\left(\tfrac{x_i - X_l}{h}\right)\delta\left(\tfrac{y_i - Y_l}{h}\right) \\ \delta(r) &= \begin{cases} \frac{1}{8}\left(3 - 2|r| + \sqrt{1 + 4|r| - 4r^2}\right), & |r| < 1 \\ \frac{1}{8}\left(5 - 2|r| - \sqrt{-7 + 12|r| - 4r^2}\right), & 1 \le |r| \le 2 \\ 0, & |r| > 2 \end{cases} \end{aligned} \tag{15}$$

where h is the mesh spacing of the Eulerian grid. The no-slip boundary condition requires that the velocity at the Lagrangian point $\boldsymbol{u}_B^{n+1}(\boldsymbol{X}_l)$ should equal the fluid velocity at the same position, which means

$$\boldsymbol{u}_B^{n+1}(\boldsymbol{X}_l) = \sum_{\boldsymbol{x}_i \in S(l)} D(\boldsymbol{x}_i - \boldsymbol{X}_l)\boldsymbol{u}^{n+1}(\boldsymbol{x}_i)h^2 \tag{16}$$

Here, $S(l)$ denotes the collection of N neighboring Eulerian cells close to the Lagrangian point l and it is defined as

$$S(l) = \left\{ i \middle| \frac{\boldsymbol{x}_i - \boldsymbol{X}_l}{h} \right| \leq 2 \right\} \tag{17}$$

Because the fluid velocity $\boldsymbol{u}^{n+1}(\boldsymbol{x}_i)$ can be computed via

$$\boldsymbol{u}^{n+1}(\boldsymbol{x}_i) = \Delta\boldsymbol{u}(\boldsymbol{x}_i) + \boldsymbol{u}^*(\boldsymbol{x}_i) \tag{18}$$

Equation (16) can be rewritten as

$$\boldsymbol{u}_B^{n+1}(\boldsymbol{X}_l) = \sum_{\boldsymbol{x}_i \in S(l)} D(\boldsymbol{x}_i - \boldsymbol{X}_l)\Delta\boldsymbol{u}(\boldsymbol{x}_i)h^2 + \sum_{\boldsymbol{x}_i \in S(l)} D(\boldsymbol{x}_i - \boldsymbol{X}_l)\boldsymbol{u}^*(\boldsymbol{x}_i)h^2 \tag{19}$$

Substituting Equation (14) into Equation (19) gives

$$\boldsymbol{u}_B^{n+1}(\boldsymbol{X}_l) = \sum_{\boldsymbol{x}_i \in S(l)} D(\boldsymbol{x}_i - \boldsymbol{X}_l)\sum_{l=1}^{M} D(\boldsymbol{x}_i - \boldsymbol{X}_l)\Delta\boldsymbol{u}_B(\boldsymbol{X}_l)\Delta s_l h^2 + \sum_{\boldsymbol{x}_i \in S(l)} D(\boldsymbol{x}_i - \boldsymbol{X}_l)\boldsymbol{u}^*(\boldsymbol{x}_i)h^2 \tag{20}$$

Clearly, the only unknown term is $\Delta\boldsymbol{u}_B(\boldsymbol{X}_l)\Delta s_l$, which determines $\Delta\boldsymbol{u}(\boldsymbol{x}_i)$ via Equation (14). Considering the whole computational domain, the above system of equations can be further written in the following matrix form:

$$\boldsymbol{A}\boldsymbol{Y} = \boldsymbol{B} \tag{21}$$

with

$$\boldsymbol{A} = \begin{bmatrix} D(\boldsymbol{x}_1 - \boldsymbol{X}_1) & D(\boldsymbol{x}_2 - \boldsymbol{X}_1) & \cdots & D(\boldsymbol{x}_N - \boldsymbol{X}_1) \\ D(\boldsymbol{x}_1 - \boldsymbol{X}_2) & D(\boldsymbol{x}_2 - \boldsymbol{X}_2) & \cdots & D(\boldsymbol{x}_N - \boldsymbol{X}_2) \\ \vdots & \vdots & \vdots & \vdots \\ D(\boldsymbol{x}_1 - \boldsymbol{X}_M) & D(\boldsymbol{x}_2 - \boldsymbol{X}_M) & \cdots & D(\boldsymbol{x}_N - \boldsymbol{X}_M) \end{bmatrix} \begin{bmatrix} D(\boldsymbol{x}_1 - \boldsymbol{X}_1) & D(\boldsymbol{x}_1 - \boldsymbol{X}_2) & \cdots & D(\boldsymbol{x}_1 - \boldsymbol{X}_M) \\ D(\boldsymbol{x}_2 - \boldsymbol{X}_1) & D(\boldsymbol{x}_2 - \boldsymbol{X}_2) & \cdots & D(\boldsymbol{x}_2 - \boldsymbol{X}_M) \\ \vdots & \vdots & \vdots & \vdots \\ D(\boldsymbol{x}_N - \boldsymbol{X}_1) & D(\boldsymbol{x}_N - \boldsymbol{X}_N) & \cdots & D(\boldsymbol{x}_N - \boldsymbol{X}_M) \end{bmatrix} \tag{22}$$

$$\boldsymbol{B} = \frac{1}{h^2}\begin{bmatrix} \boldsymbol{u}_B^{n+1}(\boldsymbol{X}_1) \\ \boldsymbol{u}_B^{n+1}(\boldsymbol{X}_2) \\ \vdots \\ \boldsymbol{u}_B^{n+1}(\boldsymbol{X}_M) \end{bmatrix} - \begin{bmatrix} D(\boldsymbol{x}_1 - \boldsymbol{X}_1) & D(\boldsymbol{x}_2 - \boldsymbol{X}_1) & \cdots & D(\boldsymbol{x}_N - \boldsymbol{X}_1) \\ D(\boldsymbol{x}_1 - \boldsymbol{X}_2) & D(\boldsymbol{x}_2 - \boldsymbol{X}_2) & \cdots & D(\boldsymbol{x}_N - \boldsymbol{X}_2) \\ \vdots & \vdots & \vdots & \vdots \\ D(\boldsymbol{x}_1 - \boldsymbol{X}_M) & D(\boldsymbol{x}_2 - \boldsymbol{X}_M) & \cdots & D(\boldsymbol{x}_N - \boldsymbol{X}_M) \end{bmatrix} \begin{bmatrix} \boldsymbol{u}^*(\boldsymbol{x}_1) \\ \boldsymbol{u}^*(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{u}^*(\boldsymbol{x}_N) \end{bmatrix} \tag{23}$$

$$\boldsymbol{Y} = \begin{bmatrix} \Delta\boldsymbol{u}_B(\boldsymbol{X}_1)\Delta s_1 \\ \Delta\boldsymbol{u}_B(\boldsymbol{X}_2)\Delta s_2 \\ \vdots \\ \Delta\boldsymbol{u}_B(\boldsymbol{X}_M)\Delta s_M \end{bmatrix} \tag{24}$$

The above system of equations implies the exact satisfaction of the no-slip boundary condition. By solving it, the unknown $\Delta\boldsymbol{u}_B(\boldsymbol{X}_l)\Delta s_l$ at all Lagrangian points can be obtained, and then $\Delta\boldsymbol{u}(\boldsymbol{x}_i)$ can be computed by Equation (14). Finally, the fluid velocity can be corrected.

In the IIBM [10,11], the matrix $\boldsymbol{A}$ is constructed, and Equation (21) is solved by computing the inverse matrix $\boldsymbol{A}^{-1}$ first. Although the computational cost for problems without

boundary movement where $\boldsymbol{A}^{-1}$ can be computed and stored first is acceptable, it could be time-consuming for cases with moving boundaries where the reassembly of $\boldsymbol{A}$ and the computation of its inversion are required at each time step. To improve the computational efficiency, the EIBM simplifies the system of equations based on the error analysis. The basic idea is to apply the Taylor series expansion to the unknown term $\Delta \boldsymbol{u}_B(\boldsymbol{X}_l)$ and Δs_l with the second-order approximation, which gives

$$\Delta \boldsymbol{u}_B(\boldsymbol{X}_l) = \Delta \boldsymbol{u}_B(\boldsymbol{X}_i) + \frac{\partial(\Delta \boldsymbol{u}_B)}{\partial \boldsymbol{X}} d\boldsymbol{X}_{li} + O\left(d\boldsymbol{X}_{li}{}^2\right) \tag{25}$$

$$\Delta s_l = \Delta s_i + \frac{\partial(\Delta s)}{\partial \boldsymbol{X}} d\boldsymbol{X}_{li} + O\left(d\boldsymbol{X}_{li}{}^2\right) \tag{26}$$

where

$$\begin{aligned} \|d\boldsymbol{X}_{li}\| &= \|\boldsymbol{X}_l - \boldsymbol{X}_i\| \\ &\leq \|\boldsymbol{x}_m - \boldsymbol{X}_i\| + \|\boldsymbol{x}_m - \boldsymbol{X}_l\| \\ &\leq \left\|(2h, 2h)^T\right\| + \left\|(2h, 2h)^T\right\| \\ &\sim O(h) \end{aligned} \tag{27}$$

This is due to the definition of the discrete delta function in Equation (15) and the limitation of the supporting region given in Equation (17) [30]. In addition, $\Delta s_i \sim O(h)$ to prevent fluid from leaking through the immersed boundary [7]. Based on the relation in Equation (13), $\Delta \boldsymbol{u}_B$ should satisfy

$$\Delta \boldsymbol{u}_B \sim O(\Delta t) = O(CFL \cdot h) = O(h) \tag{28}$$

where *CFL* denotes the Couran–Friedrichs–Lewy number. Then, $\Delta \boldsymbol{u}_B(\boldsymbol{X}_l)\Delta s_l$ can be approximated by

$$\begin{aligned} \Delta \boldsymbol{u}_B(\boldsymbol{X}_l)\Delta s_l &= \Delta \boldsymbol{u}_B(\boldsymbol{X}_i)\Delta s_i + \tfrac{\partial(\Delta \boldsymbol{u}_B)}{\partial \boldsymbol{X}} d\boldsymbol{X}_{li}\Delta s_i + \Delta \boldsymbol{u}_B(\boldsymbol{X}_i)\tfrac{\partial(\Delta s)}{\partial \boldsymbol{X}} d\boldsymbol{X}_{li} + O\left(d\boldsymbol{X}_{li}{}^2\right) \\ &= \Delta \boldsymbol{u}_B(\boldsymbol{X}_i)\Delta s_i + O\left(h^2\right) \end{aligned} \tag{29}$$

After substituting Equation (29) to Equation (20) and discarding the second-order terms $O\left(h^2\right)$, we have

$$\sum_{l \in \{A_{il} \neq 0\}} A_{il} \Delta \boldsymbol{u}_B(\boldsymbol{X}_i)\Delta s_i = \Delta \boldsymbol{u}_B(\boldsymbol{X}_i)\Delta s_i \sum_{l \in \{A_{il} \neq 0\}} A_{il} = B_i \tag{30}$$

In this way, $\Delta \boldsymbol{u}_B(\boldsymbol{X}_l)\Delta s_l$ can be solved directly without affecting the second-order accuracy of the whole solver by

$$\boldsymbol{Y} = \begin{bmatrix} \frac{1}{d_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{d_2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \frac{1}{d_M} \end{bmatrix} \boldsymbol{B} \tag{31}$$

where

$$\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_M \end{bmatrix} = \boldsymbol{D}\boldsymbol{D}^T \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{32}$$

As for the accuracy and efficiency of EIBM, the following points need to be noted. In theory, the EIBM can exactly satisfy the no-slip boundary condition, and the overall accuracy of the EIB-LBFS depends on the manner of resolving the flow filed. In this work, because the LBFS is applied to solve the flow field variables with the second-order ac-

curacy, the EIB-LBFS should be second-order accurate in space because of the accuracy limit. It is noteworthy that in practical implementation, this analytical accuracy may be degraded when the problems with moving boundaries are considered due to the difficult interpretation of the interaction between the flow filed and the moving boundaries. In the work of Zhao et al. [30], the computational complexity of the EIBM has been verified as $O(M)$, which outperforms the computational complexity of $O(M^2)$ of the IIBM. Here, the EIBM is combined with the LBFS, and this EIB-LBFS solver is embedded in the OpenFOAM framework.

3. Implementation of the EIB-LBFS Method in OpenFOAM

This section provides the implementation process of the EIB-LBFS algorithm in the OpenFOAM framework. As the LBFS is essentially a density-based solver, the developed EIB-LBFS solver mimics the main code structure of the rhoCentralFoam solver in the original OpenFOAM code.

As shown in the pseudo-code in Listing 1, following the standard operations in OpenFOAM, the regular time, FvMesh, fields, and flux objects are first created when the solver is initialized. Then, a new surfaceScalarField named streamingTime is created for storing the local streaming time δ_t variable at each interface of the cells in the mesh. An LBFS object of the user-defined lbfsModule class is further initialized, where the value of δ_t is precomputed and the discrete velocity model of the local LBFS is pre-constructed at each cell interface (according to Section 2.2). In a similar fashion, the IBM object (of the user-defined ibmModule class) handles the EIBM calculations according to the description in Section 2.3. In the constructor of the IBM object, the properties of the immersed boundary, such as the spacing of the Eulerian mesh h, the coordinates of the Lagrangian points, and the prescribed motion or the kinematic laws of the immersed object (if applicable), are loaded into the solver.

Up to this point, the solver initialization is finished, and the main loop for marching the solution in time is started. At the beginning of each time step, the continuity and momentum fluxes are computed by calling the calcFlux function of the LBFS object (according to Equation (7)). Then, the divergence of the fluxes is computed using the built-in fvc::div method in OpenFOAM, and the intermediate density and momentum fields are updated from Equation (4). The intermediate velocity filed can be further obtained. A new vectorField deltaU is created for the velocity correction $\Delta \boldsymbol{u}$, and its value is computed by calling the getDeltaU member function of the IBM object, which updates the $\boldsymbol{D}$ and $\boldsymbol{B}$ matrices in Equations (31) and (32) and solves the explicit algebraic equations. After the velocity field is updated, the new momentum fields can be corrected, which finishes the computations of one time step. Finally, the above process is repeated until the convergence conditions are satisfied.

Listing 1. The pseudo-code of the developed EIB-LBFS solver in OpenFOAM.

```
int main(int argc, char *argv[])
{
    // standard operations in OpenFOAM
    #include “setRootCase.H”
    #include “createTime.H”
     #include “createMesh.H”
    #include “createFields.H”
    #include “createFieldRefs.H”
     #include “createFluxes.H”

    // initialize streaming time variable for local LBFS reconstruction
      surfaceScalarField streamingTime;
```

```
    // create LBFS module object for handling interfacial flux calculation according to Section 2.2
      lbfsModule LBFS(mesh, streamingTime, rho, U);

    // create IBM module object for handling EIBM calculation according to Section 2.3
      ibmModule IBM(U, rho);

    // start main solution loop
    Info<< "\nStarting time loop\n" << endl;
    while (runTime.run())
    {
        runTime++;
        Info<< "Time = " << runTime.timeName() << nl << endl;
        // call the member function of the LBFS object to compute the fluxes, where mu, phi and
phiUp denote the dynamic viscosity, mass flux and momentum flux, respectively.
        LBFS.calcFlux(mu, phi, phiUp);

        volScalarField rhoR = -fvc::div(phi); // rho residual as the divergence of the continuity
flux
        volVectorField rhoUR = -fvc::div(phiUp); // rhoU residual as the divergence of the
momentum flux
        // solve intermediate continuity and momentum equations according to Equation (4)
        solve(fvm::ddt(rho)==rhoR);
        solve(fvm::ddt(rhoU)==rhoUR);

        // get intermediate velocity field
        U.ref() = rhoU.internalField()/rho.internalField();

        // call the member function of the IBM object to compute the velocity correction to the
intermediate velocity according to Section 2.3
        vectorField deltaU = IBM.getDeltaU();
        // correct the velocity in the Eulerian mesh
        forAll (mesh.C(), i)
            {
                U.ref()[i] = U.internalField()[i] + deltaU[i];
            }

        // update the momentum based on the new velocity field
        rhoU = rho*U;

        // update boundary conditions at domain boundaries
        U.correctBoundaryConditions();
        rho.correctBoundaryConditions();
        rhoU.ref() = U.internalField()*rho.internalField();
        rhoU.boundaryFieldRef() = rho.boundaryField()*U.boundaryField();

        // standard operations in OpenFOAM

        runTime.write();
    }
    Info<< "End\n" << endl;
    return 0;
}
```

4. Numerical Results and Discussion

This section tests a series of benchmark cases to assess the performance of the EIB-LBFS implementation in OpenFOAM. The overall accuracy is first tested by solving the Taylor–Green decaying vortex problem. The following case is the flow past a stationary cylinder, and then the flow past a NACA-0012 airfoil, which has a more complicated geometry, is studied. To further validate the capability of the present solver to resolve

flow problems with a moving boundary, the flow past an oscillating circular cylinder is tested. In these three tests, the immersed boundary method, i.e., the direct forcing method, in FOAM-extend version 4.0 is also applied for comparison. For simplicity, the EIB-LBFS method is termed the "present" method and the IBM in FOAM-extend version 4.0 is termed "FoamExtend" in the following comparison. Lastly, the present EIB-LBFS solver is applied to simulate an FSI problem, namely, the sedimentation of a circular particle in a rectangular box.

4.1. Accuracy Test According to the Taylor–Green Vortex

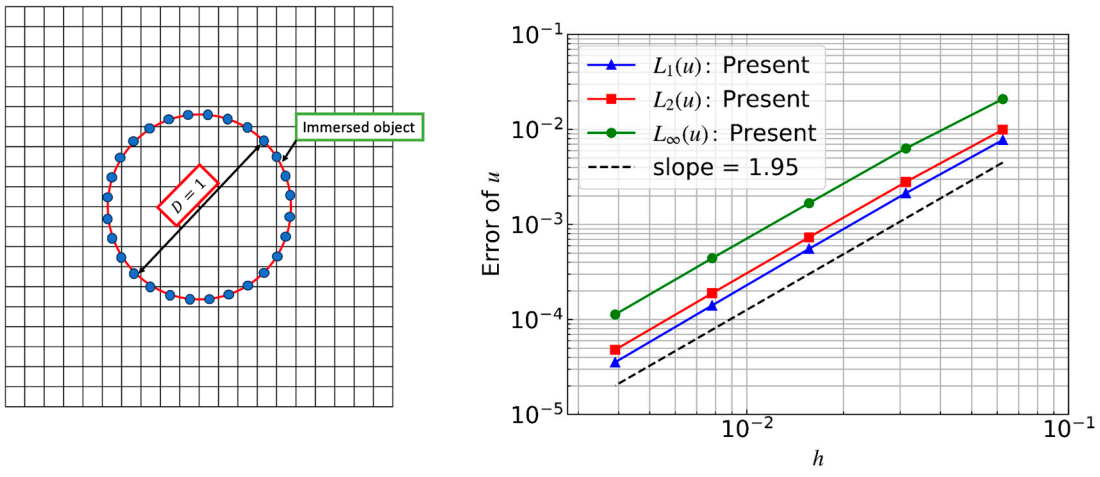
The decaying vortex flow problem [11,40], which has an analytic solution, is used to test the overall accuracy of the EIB-LBFS method numerically. The analytical solution of the problem satisfying the 2D incompressible N-S equations reads

$$\begin{aligned} u(x,y,t) &= -U\cos(\pi x/L)\sin(\pi y/L)e^{-2\pi^2 Ut/(\mathrm{Re}L)} \\ v(x,y,t) &= U\sin(\pi x/L)\cos(\pi y/L)e^{-2\pi^2 Ut/(\mathrm{Re}L)} \\ \rho(x,y,t) &= \rho_0 - \frac{\rho_0 U^2}{4{c_s}^2}[\cos(2\pi x/L)+\cos(2\pi y/L)]e^{-4\pi^2 Ut/(\mathrm{Re}L)} \end{aligned} \tag{33}$$

Numerical simulations are conducted on the computational domain of $[-L,L]\times[-L,L]$ with periodic boundary conditions at a Reynolds number of $Re = UL/\nu = 10$. The relaxation parameter τ is set as 0.8 and ρ_0 is taken as 1. As shown in Figure 1a, a circle with a diameter of D = 1.0 is immersed in the domain. To test the spatial accuracy of the present EIBM, the analytical solution is enforced on the cylinder surface. The flow is also initialized from the analytical solution at t = 0. The solutions on the Eulerian mesh at $t = L/U = 1$ are computed, and the relative errors of velocity component u are measured using the L_∞, L_1, and L_2 norms, which are defined as

$$L_\infty(u) = \max_{1\le i\le Ncell}\left(\left|\frac{u_i-u_i^e}{U}\right|\right), L_1(u) = \frac{1}{Ncell}\sum_{i=1}^{Ncell}\left(\left|\frac{u_i-u_i^e}{U}\right|\right), L_2(u) = \left(\frac{1}{Ncell}\sum_{i=1}^{Ncell}\left(\frac{u_i-u_i^e}{U}\right)^2\right)^{\frac{1}{2}} \tag{34}$$

where u_i and u_i^e represent the numerical result and the exact solution, respectively. *Ncell* is the number of the cells.



(**a**) (**b**)

Figure 1. Spatial accuracy test of the EIB-LBFS for the Taylor–Green vortex: (**a**) schematics of the simulation and (**b**) convergence history of errors with mesh refinement.

For the convergence study, the regular uniform grids with a spacing of h = 1/8 to 1/128 are used. Relative errors of velocity component u and the rates of convergence are

shown in Figure 1b. As can be seen in the figure, the EIB-LBFS method can achieve the second-order accuracy for the problem with the immersed boundaries, which is consistent with the above theoretical discussion of accuracy.

4.2. Flow Past a Stationary Circular Cylinder

First, the benchmark case of flow past a stationary circular cylinder [37,40,43] is simulated to examine whether the present solver can exactly satisfy the no-slip boundary condition. In this case, the incoming viscous fluid with a free-stream velocity U_0 flows past a fixed circular cylinder. The Reynolds number characterizing the flow pattern is given as $Re = U_0 D_c/\nu$ with the diameter of the circular cylinder D_c. In the simulations, the cases of Re = 20, 40, 100 and 200 are tested.

The pressure coefficient C_p, lift coefficient C_l, drag coefficient C_d, and Strouhal number S_t are used to quantify the numerical results, and they are defined as follows:

$$C_p = \frac{p_w - p_0}{\rho_0 U_0^2/2},\ C_l = \frac{F_l}{\rho_0 U_0^2/2},\ C_d = \frac{F_d}{\rho_0 U_0^2/2},\ S_t = \frac{f_o L}{U_0} \tag{35}$$

where p_0 and p_w denote the pressure of the free-stream and on the cylinder surface, respectively. F_l and F_d are, respectively, the lift force and the drag force. f_o represents the vortex shedding frequency.

A no-slip boundary condition is applied on the cylinder surface, and the far-field free-stream condition is enforced on the outer boundary. Figure 2 shows the computational domain and the hybrid unstructured mesh is used for all four cases, with 59,973 cells in total. In the region of 0.8 D_c × 1.0 D_c around the circular cylinder, a uniform mesh of the mesh spacing h = 0.01 D_c is used. The outer computational domain boundaries are located at 50 D_c away from the cylinder. On the surface of the circular cylinder, there are 314 uniformly distributed Lagrangian points.

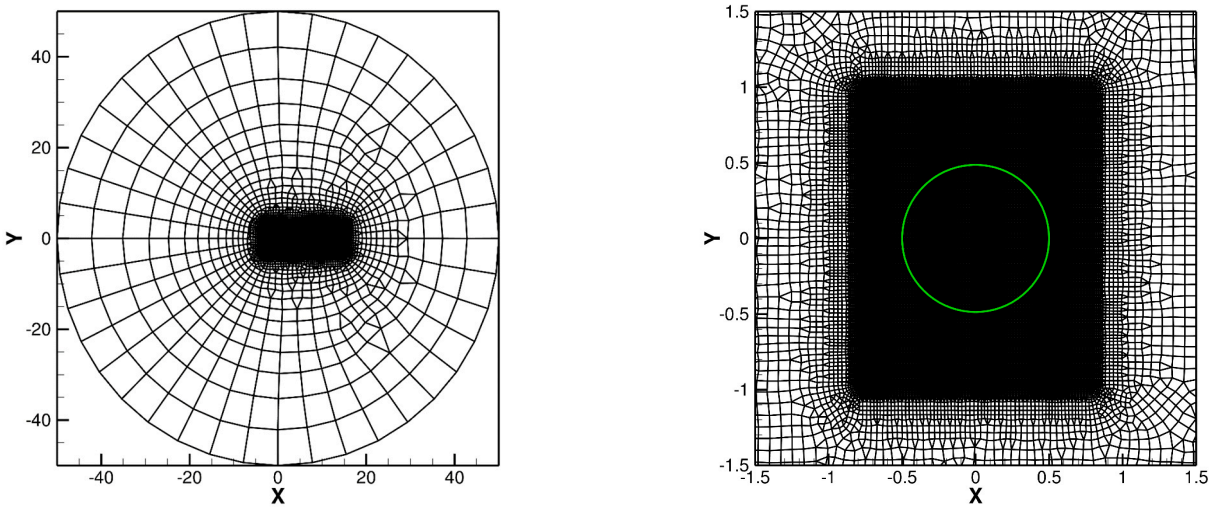


Figure 2. Entire unstructured mesh (**left**) and close-up view of the uniform mesh around the cylinder (**right**) for viscous flow past a circular cylinder. The cylinder boundary is represented by the green circle.

As shown in Figure 3, the present EIB-LBFS method can obtain the correct flow fields, which agree well with those in studies [37,40,43,44] for both cases of Re = 20 and 40. In addition, the streamlines have no unphysical penetration, which verifies the guarantee of the no-slip boundary condition by the EIB-LBFS method. To compare with the IBM in FOAM-Extend 4.0, Figure 4 compares the velocity magnitude $\|U\|$ versus the azimuth angle θ on the solid boundary. Clearly, the IBM in FOAM-Extend has one order of magnitude larger velocity magnitude than the present method on the same mesh. Such an observation

confirms that the IBM in FOAM-Extend cannot enforce the no-slip boundary condition as accurately as the present method. This conclusion is further proved by the comparison of the pressure coefficient C_p in the case of $Re = 40$ shown in Figure 5, where the surface pressure distribution computed by the FOAM-Extend 4.0 exhibits spurious oscillations. The better agreement of the result of the present method with the reference data [49] validates the more exact satisfaction of the no-slip boundary condition for the present EIB-LBFS method than the IBM in FOAM-Extend 4.0. For a quantitative comparison, Table 1 tabulates the drag coefficient C_d and the geometrical quantities of the eddies, namely, the recirculation length L_s and the separation angle θ_s obtained by both methods with some reference data [10,39,50–53]. It can be seen that, using the same mesh, the present method predicts good results within the range of the reference data, while the results from the IBM in FOAM-Extend have larger deviations due to the inaccurate satisfaction of the no-slip wall boundary condition.

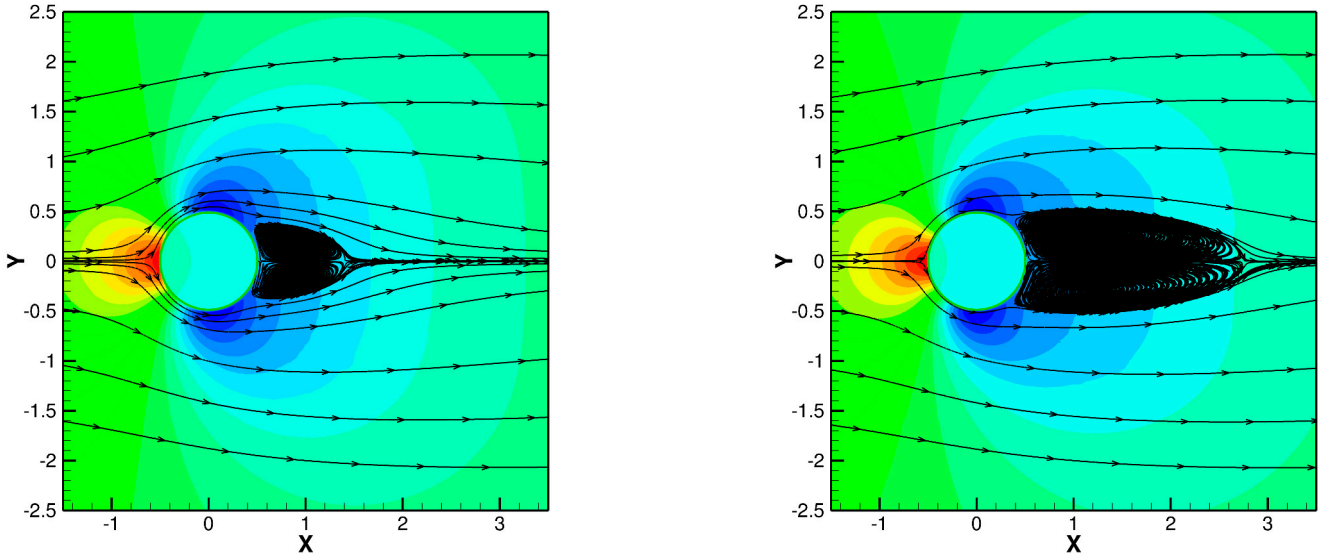


Figure 3. Pressure contours and streamlines around the cylinder obtained using the EIB-LBFS method for flow past a cylinder at $Re = 20$ (**left**) and $Re = 40$ (**right**).

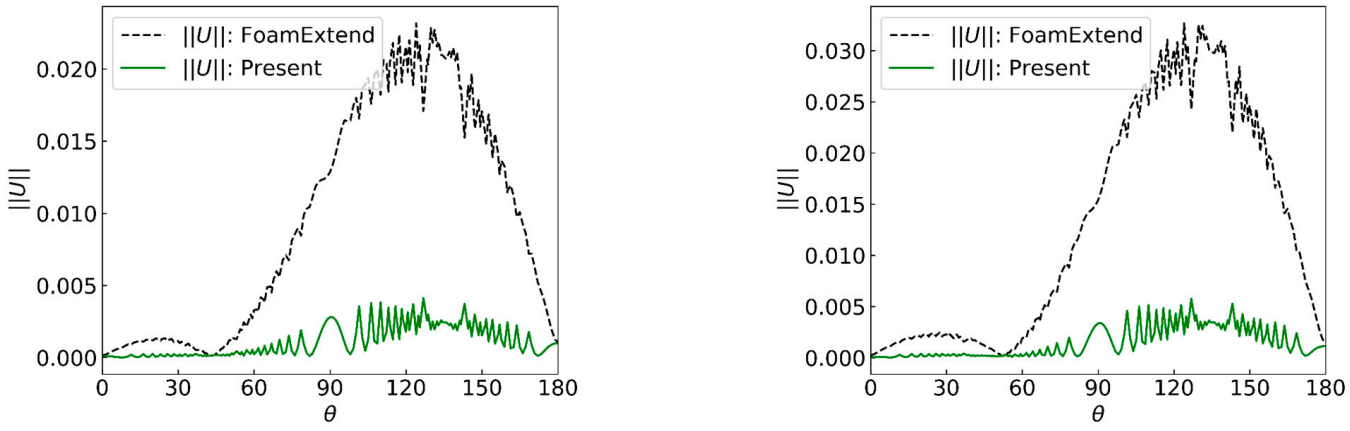


Figure 4. Comparison of the velocity on the cylinder surface obtained using the EIB-LBFS method and the IBM in FOAM-Extend for flow past a cylinder at $Re = 20$ (**left**) and $Re = 40$ (**right**).

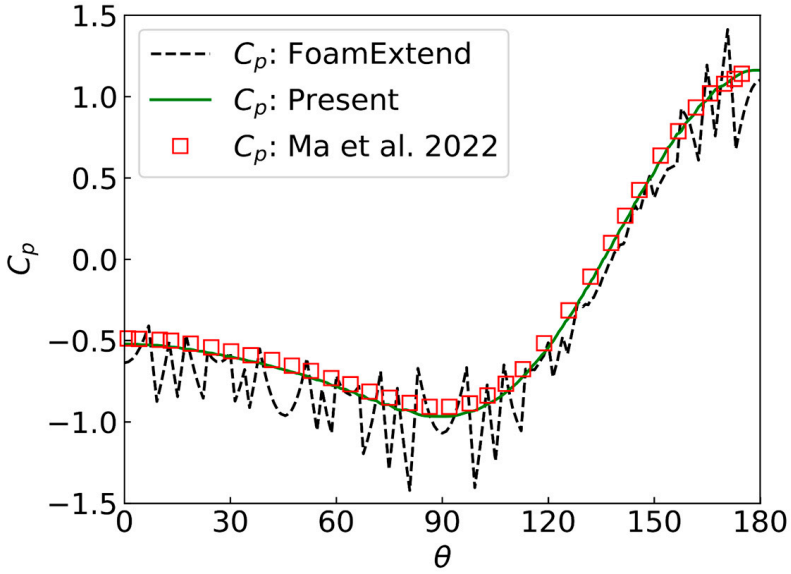


Figure 5. Comparison of pressure coefficient on the cylinder wall obtained using the EIB-LBFS method and the IBM in FOAM-Extend with the reference data from Ma et al. [49] for flow past a cylinder at Re = 40.

Table 1. Comparison of drag coefficient, recirculation length, and separation angle for steady flow past a stationary circular cylinder at Re = 20 and 40.

Re	References	C_d	L_s/D_c	θ_s
20	Dennis and Chang [50]	2.05	0.94	43.7
	Shukla et al. [51]	2.07	0.92	43.3
	Wu and Shu et al. [10]	2.091	0.93	-
	EIB-LBFS	2.05	0.92	43.32
	FOAM-Extend	2.18	0.97	47.31
40	He and Doolen [52]	1.499	2.245	52.84
	Pellerin et al. [53]	1.505	2.259	53.64
	Shu et al. [39]	1.53	2.24	52.69
	EIB-LBFS	1.531	2.254	52.54
	FOAM-Extend	1.633	2.252	54.88

For the cases of Re = 100 and 200, the flow is unsteady, and vortex shedding occurs in the wake region due to the small physical viscosity. This phenomenon can be seen in Figure 6. Figure 7 compares the evolution of the drag coefficient C_d and lift coefficient C_l computed using the two methods. There is an obvious difference between these results, and the quantitative comparison with the reference data [39,40,54–58] is given in Table 2. The Strouhal number St is also included to quantify the frequency of the vortex shedding. As can be seen, the results of the EIB-LBFS method have better agreement with the reference data than those computed by the IBM in FOAM-Extend 4.0 on the same mesh. This observation validates the reliability of the present method for solving such an unsteady problem with curved geometry on hybrid unstructured grids. In addition, it is found that the present method outperforms the IBM embedded in FOAM-Extend 4.0 in terms of accuracy.

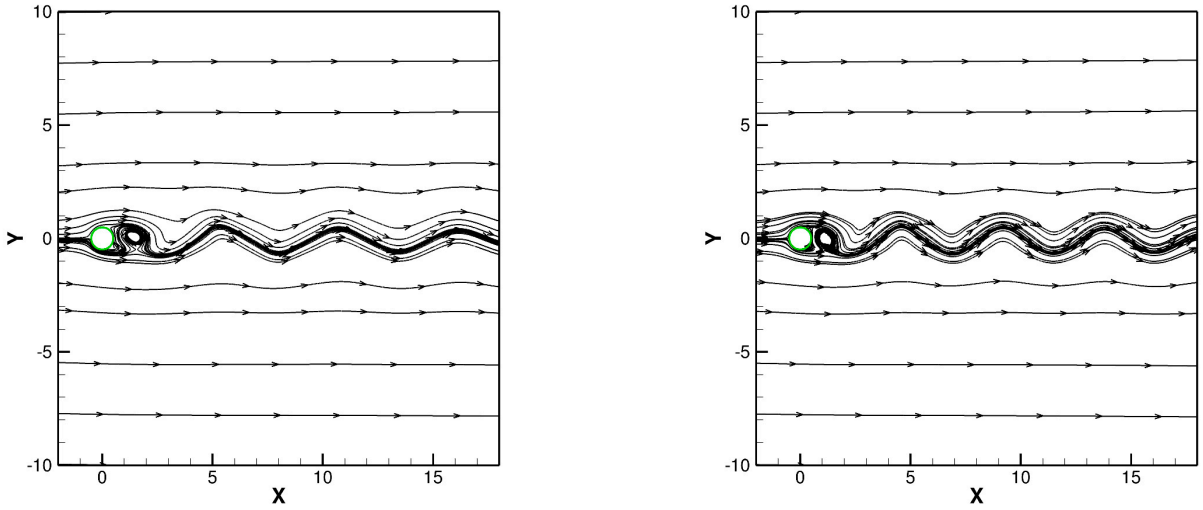


Figure 6. Streamlines around the cylinder obtained using the EIB-LBFS method for flow past a cylinder at *Re* =100 (**left**) and *Re* =200 (**right**).

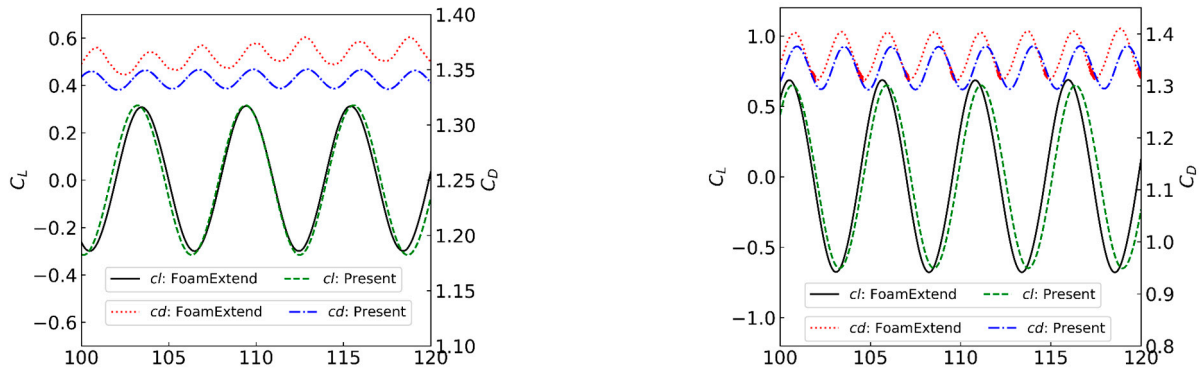


Figure 7. Comparison of evolution of the lift and drag coefficients obtained using the EIB-LBFS method and the IBM in FOAM-Extend for flow past a cylinder at *Re* = 100 (**left**) and *Re* = 200 (**right**).

Table 2. Comparison of dynamic parameters for unsteady flow past a stationary circular cylinder at *Re* = 100 and 200.

Re	References	C_l	C_d	*St*
100	Braza et al. [54]	±0.30	1.28±0.02	0.16
	Liu et al. [55]	±0.339	1.350 ± 0.02	0.164
	Shu et al. [39]	±0.33	1.334 ± 0.02	0.164
	Liu et al. [40]	±0.332	1.337 ± 0.011	0.164
	Pellerin et al. [53]	±0.325	1.325	0.164
	EIB-LBFS	±0.316	1.341 ± 0.01	0.161
	FOAM-Extend	±0.303	1.359 ± 0.02	0.167
200	Posdziech and Grundmann [56]	±0.673	1.325	0.195
	Persillon and Braza [57]	-	1.321	0.198
	Franke et al. [58]	±0.65	1.31	0.194
	EIB-LBFS	±0.650	1.335 ± 0.04	0.191
	FOAM-Extend	±0.690	1.359 ± 0.05	0.194

4.3. Flow Past a Stationary NACA-0012 Airfoil

To further assess the performance of the present solver for the more complicated configuration, the flow past a stationary NACA-0012 airfoil is simulated. Following the setting in Ref. [10], the Reynolds number defined as $Re = U_0 L/\nu$ is set as 500, where L denotes

the chord length of the airfoil. The attack angle of 0° is considered. The computational domain of $[-50\ L, 50\ L] \times [-50\ L, 50\ L]$ discretized by 63,415 hybrid unstructured cells is presented in Figure 8, where the uniform mesh of the grid spacing $h = 0.005\ L$ is used in the region of $[-0.2\ L, 1.4\ L] \times [-0.15\ L, 0.15\ L]$ around the airfoil. The surface of the airfoil is represented by 412 uniformly distributed Lagrangian points. Figure 9 plots the pressure contours around the airfoil computed using the EIB-LBFS method and the IBM in FOAM-Extend 4.0. Both methods can obtain the symmetric flow pattern, but the result of the FOAM-Extend shows clear staircase patterns adjacent to the airfoil due to the cutout of the solid cells. Figure 10 further compares the u-velocity contours and streamlines around the airfoil calculated using these two methods. Although the velocity contours basically agree, it is obvious that the streamlines of the EIB-LBFS method do not penetrate the solid boundary, while minor penetrations are detected in the results computed using FOAM-Extend. The comparison of u-velocity and v-velocity profiles at various cross sections, as shown in Figure 11, shows good agreement with the reference data [59] for the present method, which indicates the exact satisfaction of the no-slip boundary condition for the present method. Furthermore, the computed C_d by the present method is 0.1750, which agrees with the reference value of 0.1762 [59]. Thus, the capability and accuracy of the present method are well validated.

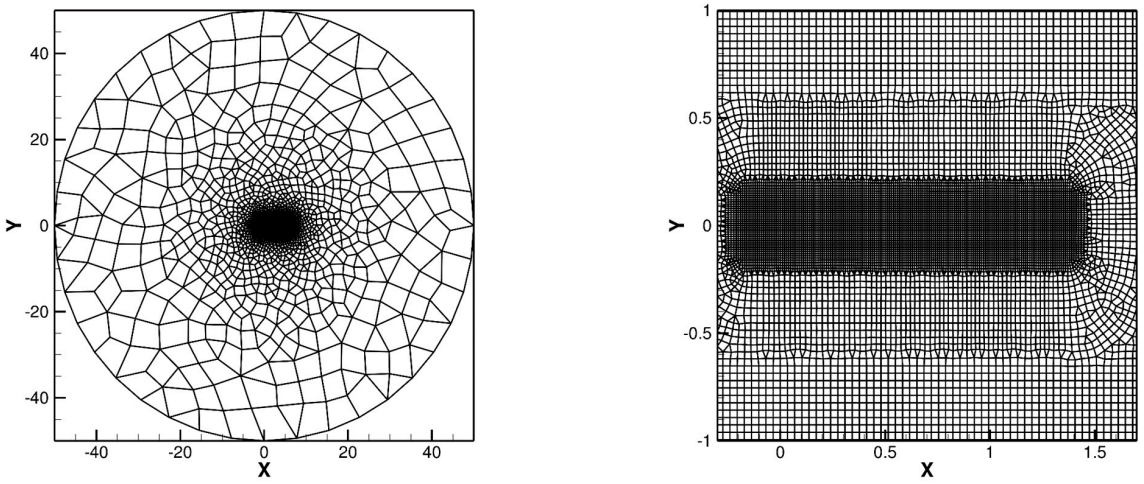


Figure 8. Entire unstructured mesh (**left**) and close-up view of the uniform mesh around the airfoil (**right**) for flow past a stationary NACA-0012 airfoil at $Re = 500$.

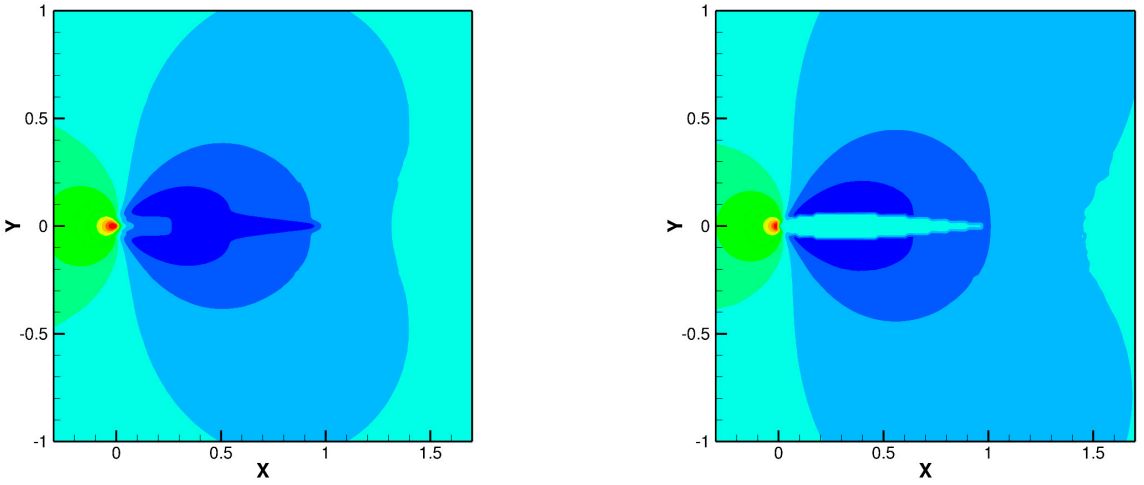


Figure 9. Comparison of pressure contours around the airfoil obtained using the EIB-LBFS method (**left**) and the IBM in FOAM-Extend (**right**) for flow past a stationary NACA-0012 airfoil at $Re = 500$.

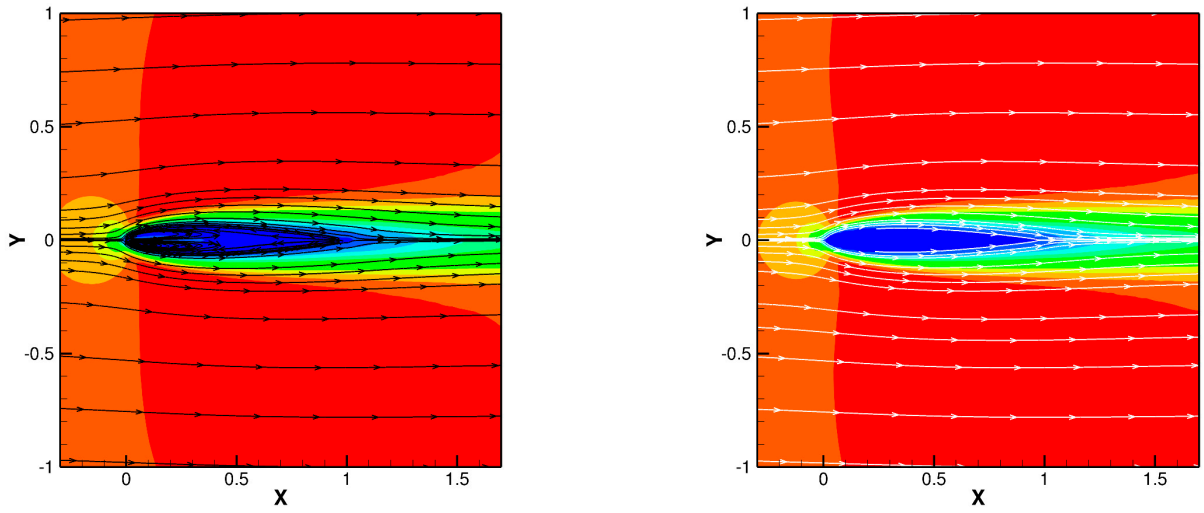
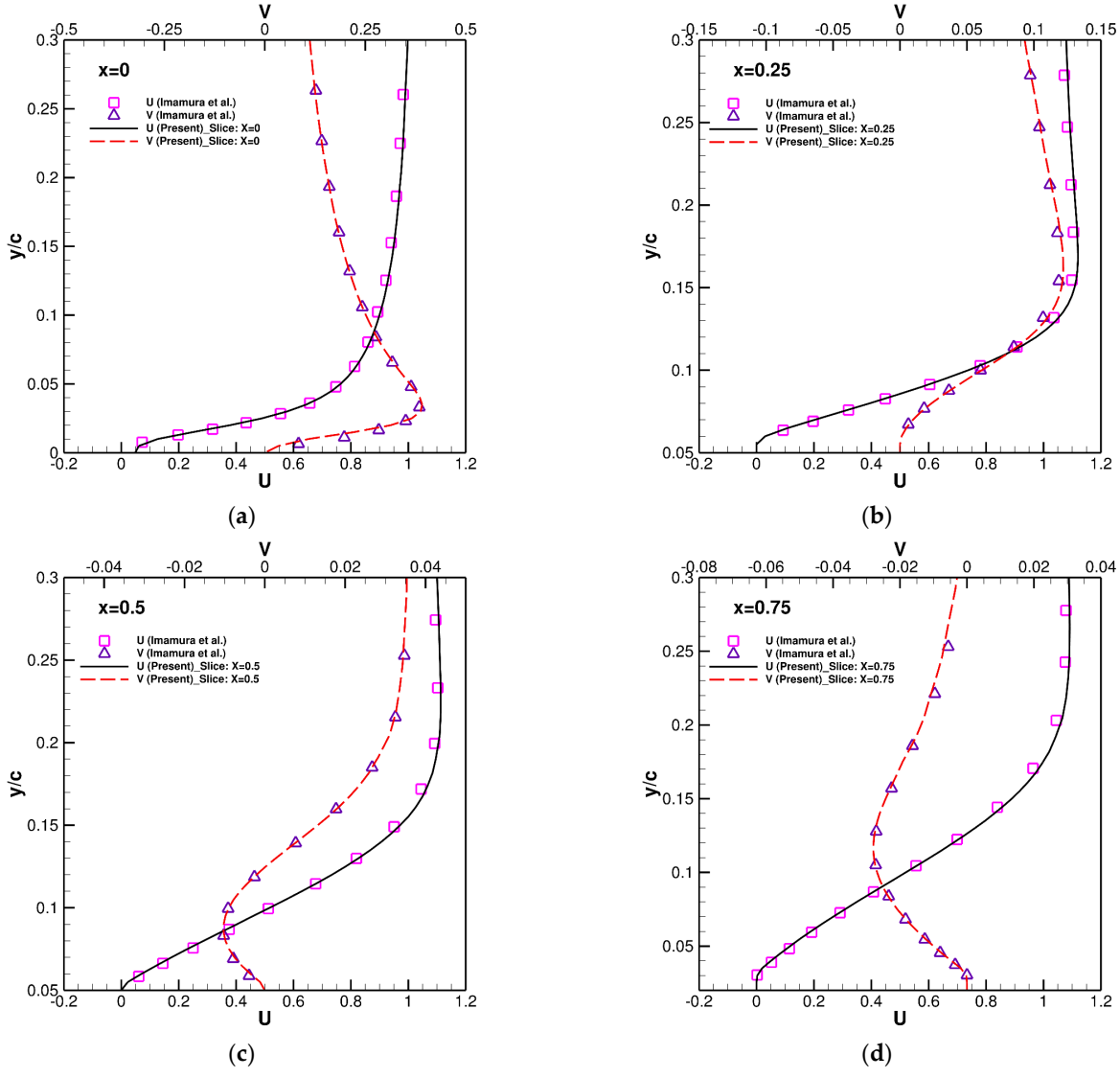


Figure 10. Comparison of u-velocity contours and streamlines around the airfoil obtained using the EIB-LBFS method (**left**) and the IBM in FOAM-Extend (**right**) for flow past a stationary NACA-0012 airfoil at Re = 500.



(**a**) (**b**) (**c**) (**d**)

Figure 11. *Cont.*

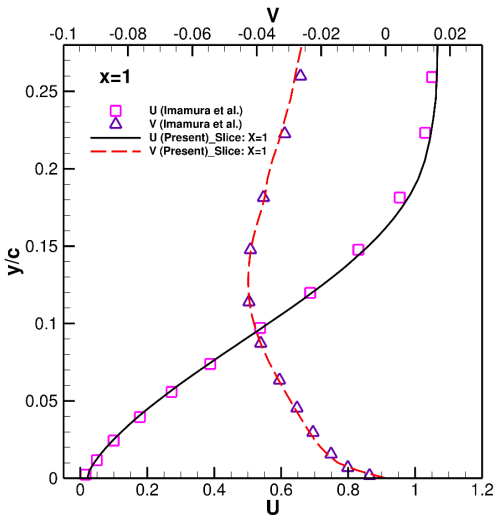


Figure 11. Comparison of u-velocity and v-velocity profiles at (**a**) $x = 0.00$, (**b**) $x = 0.25$, (**c**) $x = 0.50$, (**d**) $x = 0.75$, and (**e**) $x = 1.00$ obtained using the EIB-LBFS method with reference data reported by Imamura et al. [59] for flow past a stationary NACA-0012 airfoil at $Re = 500$.

4.4. Flow Past an Oscillating Circular Cylinder

After validating the good performance of the present solver for problems with stationary boundaries, the assessment of problems with boundary movement is further conducted. Here, the uniform flow past an oscillating circular cylinder [60] is investigated. Similarly to the first case in Section 4.2, this problem is characterized by the Reynolds number $Re = U_0 D_c/\nu$ with the free-stream velocity U_0 and the circular cylinder dimeter D_c. In this case, a transverse oscillation motion is forced, and the instantaneous position of the cylinder is given as

$$y(t) = A_e \sin(2\pi t f_e) \tag{36}$$

where A_e denotes the oscillating amplitude and f_e is the excitation frequency. In the simulation, the parameters are set as $Re = 185$, $A_e/D_c = 0.2$ and $f_e/f_o = 0.8$, 0.9, 1.0, 1.1, and 1.2, where f_o denotes the natural vortex shedding frequency of the cylinder at this Reynolds number. The same mesh as shown in Figure 2 is used for all computations.

To quantify the numerical results of the present EIB-LBFS method, the time-averaged drag coefficient "*cd mean*" and the root-mean-square value "*cd rms*" are compared in Figure 12. Clearly, they agree well with the data from Refs. [30,37]. Figure 13 displays the evolutions of the drag coefficient C_d and lift coefficient C_l for different ratios of frequencies. For the cases of $f_e/f_o = 0.8$, 0.9, and 1.0, the evolutions of C_d and C_l are harmonic, and the amplitude increases while the period decreases as f_e/f_o becomes larger. Compared to the results of the present method, both force coefficients obtained using the IBM in FOAM-Extend 4.0 have clear non-physical oscillations. This observation demonstrates that the present EIB-LBFS method can satisfy the no-slip boundary conditions more accurately than the IBM in FOAM-Extend 4.0. When f_e/f_o is larger than 1.0, i.e., $f_e/f_o = 1.1$ and 1.2, the amplitude modulation for the force coefficients can be seen. In addition, there are many differences in the phase and amplitude between the results of the two methods.

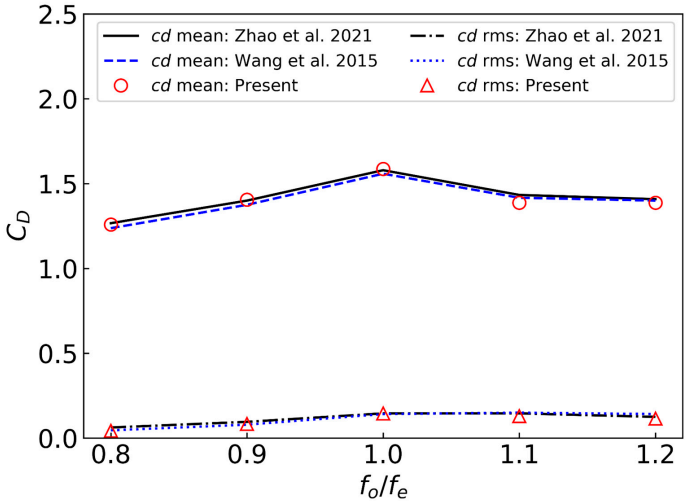


Figure 12. Comparison of the computed mean drag coefficient and root-mean-square value of drag coefficient with the reference data from Wang et al. [37] and Zhao et al. [30] for flow past an oscillating circular cylinder.

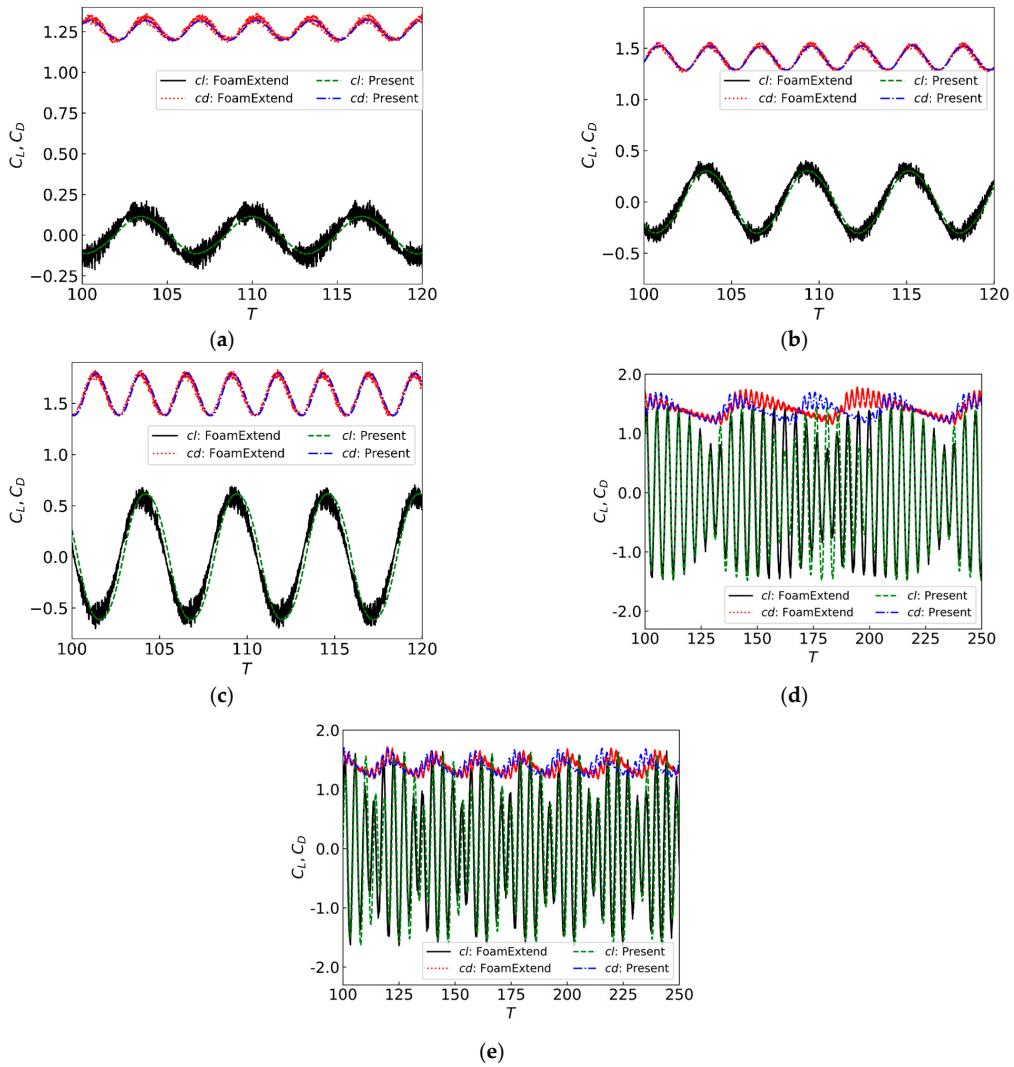


Figure 13. Comparison of lift and drag coefficients for flow past an oscillating circular cylinder: (**a**) $f_e/f_o = 0.8$, (**b**) $f_e/f_o = 0.9$, (**c**) $f_e/f_o = 1.0$, (**d**) $f_e/f_o = 1.1$, and (**e**) $f_e/f_o = 1.2$.

4.5. Sedimentation of a Circular Particle in a Rectangular Box

In this subsection, the present solver is applied to simulate the sedimentation of a circular particle in a rectangular domain [20,61]. As an FSI problem, the uncertainty of the body trajectories, the long boundary movement, and the tedious re-meshing process cause difficulty for the conventional methods using the body-fitted meshes. Thus, it is a good problem to evaluate the accuracy and reliability of the present solver. As shown in Figure 14, the computational domain is a rectangular box for which the width is 2 cm and the length is 6 cm. A rigid circular cylinder with a diameter of D_c = 0.25 cm is statically placed at the location of (1 cm, 4 cm) in the box. The density of the particle is ρ_s = 1.25 g/cm^3, and the density of the static fluid is ρ_f = 1.0 g/cm^3. The dynamic viscosity of the fluid is 0.1 g/(cm·s). Hybrid unstructured mesh is used to discretize the computational domain. In the region along the sedimentation trajectory, the uniform mesh of h = 0.02 D_c is applied as displayed in Figure 14. There are 225 Lagrangian points on the surface of the particle. All boundaries of the box are no-slip walls.

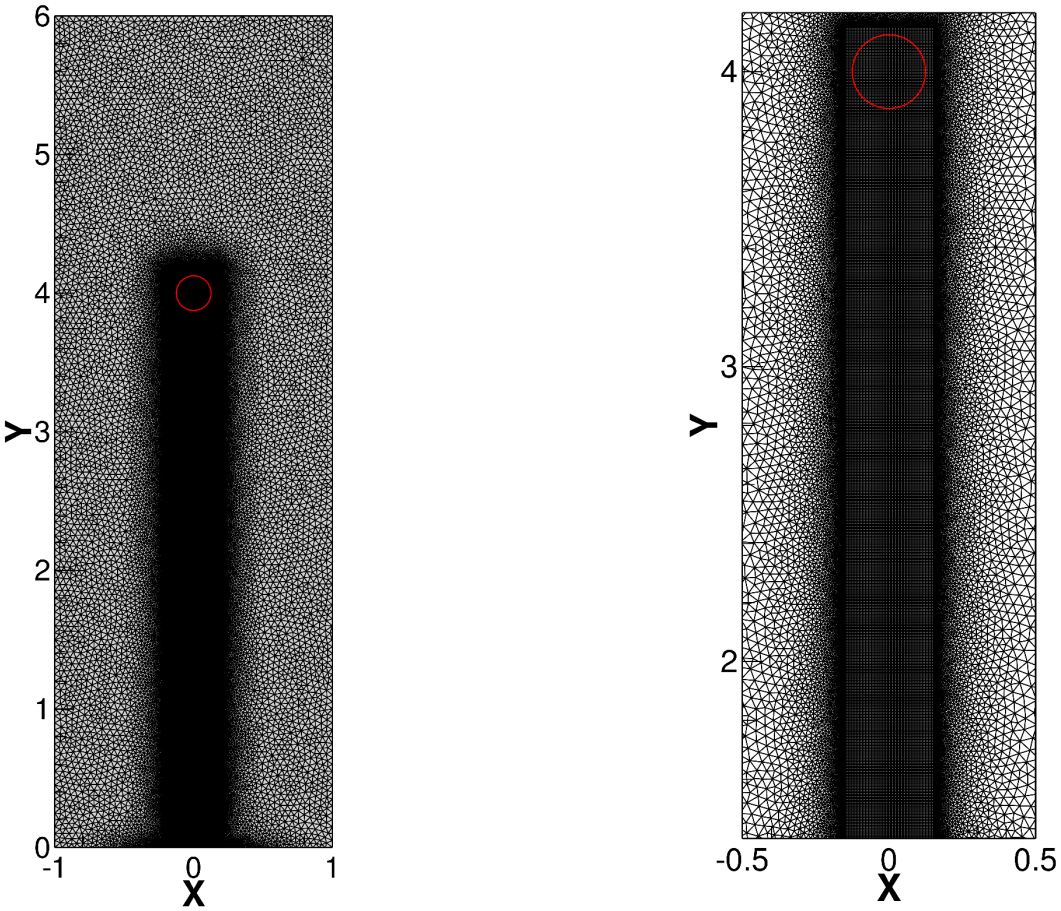


Figure 14. Entire mesh (**left**) and close-up view of mesh in the region along the sedimentation trajectory (**right**) for sedimentation of a circular particle in a rectangular box. The solid boundary of the circular particle is represented by the red circle.

After releasing the particle at t = 0 s, the particle will freely fall under the effect of gravity and the resistance from the fluid. Figure 15 plots the instantaneous vorticity contours at four time instants of t = 0.1 s, 0.3 s, 0.5 s, and 0.7 s obtained using the present EIB-LBFS method, which shows good agreement with the numerical results in Ref. [30]. Figure 16 compares the computed evolutions of the vertical coordinate Y, vertical velocity V, and local Reynolds number Re with the reference data [30,62], thus confirming excellent agreement. These observations validate that the EIB-LBFS method can exactly satisfy the no-slip boundary condition and provide accurate results for the FSI problem.

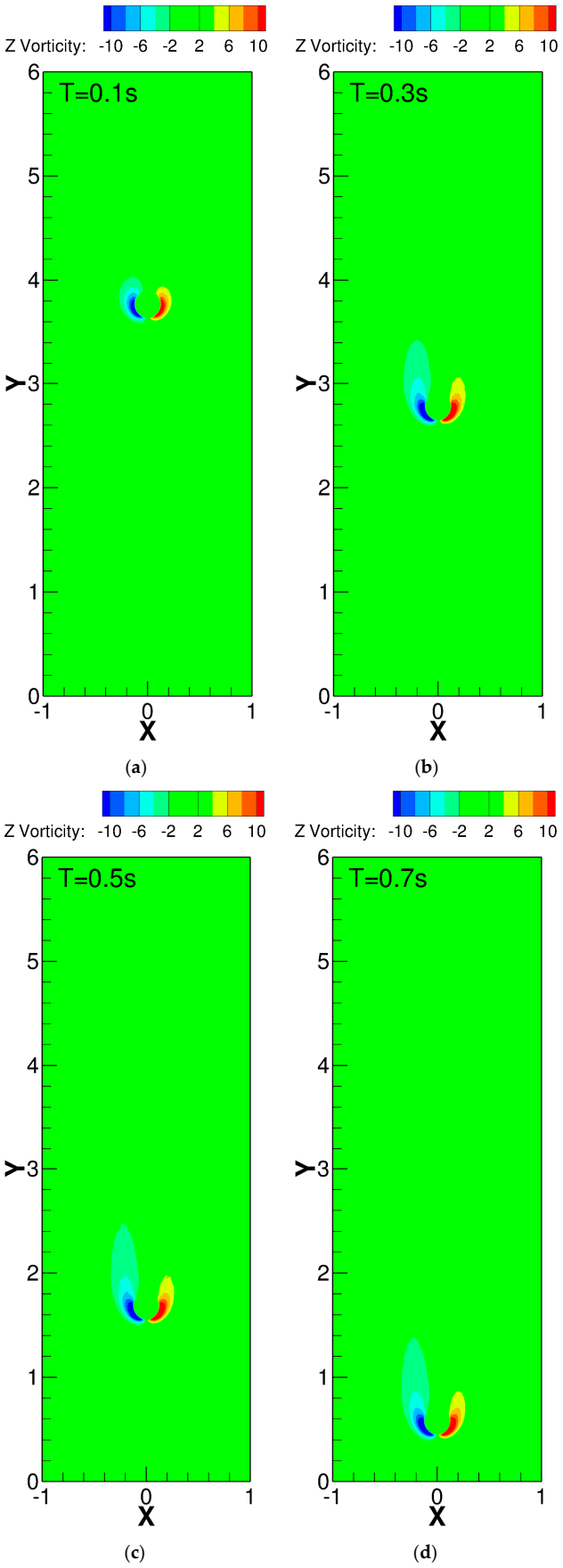


Figure 15. Vorticity contours at (**a**) $t = 0.1$ s, (**b**) $t = 0.3$ s, (**c**) $t = 0.5$ s, and (**d**) $t = 0.7$ s obtained using the EIB-LBFS method for the falling circular cylinder in a rectangular box.

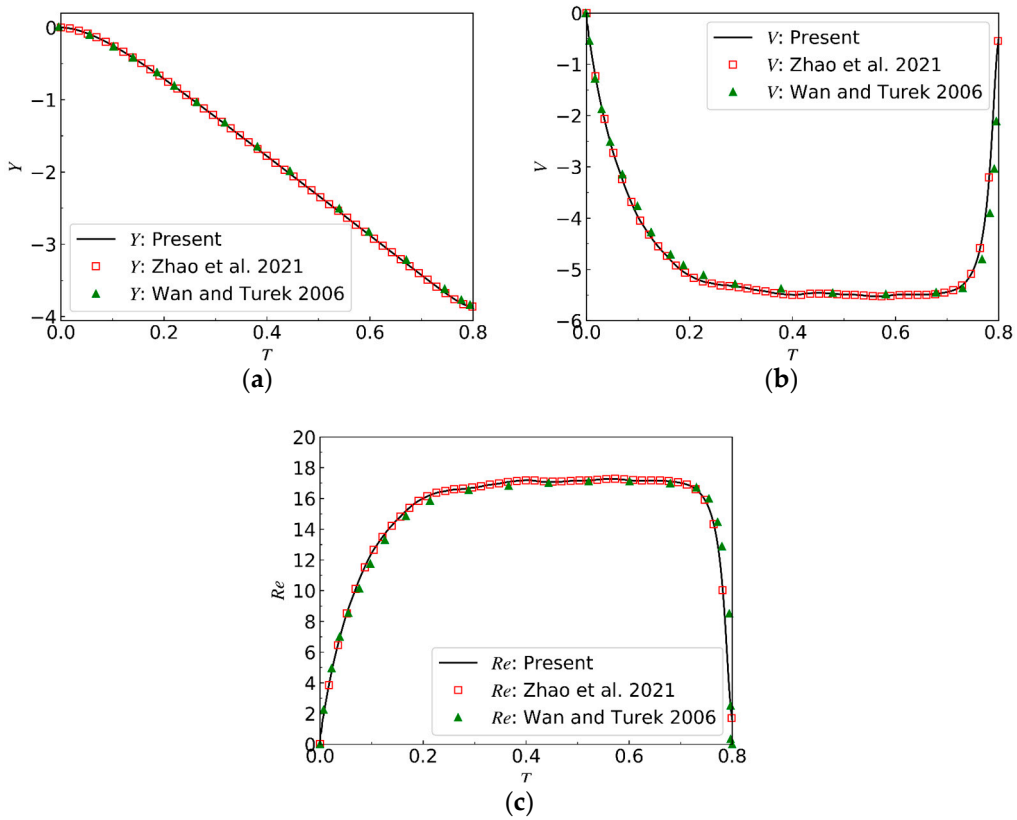


Figure 16. Comparison of the time evolution of (**a**) vertical coordinate, (**b**) vertical velocity, and (**c**) local Reynolds number obtained using the EIB-LBFS method with reference data from Zhao et al. [30] and Wan and Turek [62] for sedimentation of a circular particle in a rectangular box.

5. Conclusions

Flow problems with moving boundaries and fluid-structure interactions are very common in industries and in nature. IBM has become one popular CFD method to solve such problems. However, for incompressible flow problems with stationary or moving boundaries and FSI problems, the unphysical streamline penetration, low computational efficiency, and flexibility of using unstructured grids are key problems. To address these problems, this work combines the explicit boundary-condition-enforced immersed boundary method with the lattice Boltzmann flux solver and integrates the resulting EIB-LBFS into the open-source OpenFOAM platform for effective simulation of incompressible flow problems with moving boundaries and fluid–structure interaction problems. Because the original boundary-condition-enforced IBM applies the implicit technique to solve the linear equation system for the velocity correction, it could be inefficient when the large-scale problems or moving boundary are considered. By computing the velocity correction explicitly, the EIBM improves the computational efficiency with a global second-order accuracy in space. In addition, the LBFS inherits the advantages of the LBM for solving the incompressible viscous flow problems and avoids the drawbacks of the LBM, such as the lattice uniformity, and the tie-up between the time step and the lattice spacing. The LBFS can simultaneously evaluate the inviscid and viscous fluxes at the cell interface, which is more straightforward than the separate method, like the Riemann solver for the inviscid flux and the central finite difference for the viscous flux. By incorporating the EIB-LBFS method into the OpenFOAM, it could be much easier to apply this effective method to solve various flow problems with moving boundaries and FSI problems.

To assess whether the no-slip boundary condition can be exactly satisfied or not in the EIBM and to evaluate the ability and accuracy of the EIB-LBFS method for solving problems with complex geometries and moving boundaries, a series of representative cases are tested. First, the accuracy test validates that the EIB-LBFS is second-order accurate in

space. By solving the problems of flow past a stationary circular cylinder and NACA0012 airfoil, the exact satisfaction of the no-slip boundary condition and the good accuracy for the EIB-LBFS method are then verified via the good agreement between the obtained results and the reference data. The good performance of the present method is further validated by solving well the flow past an oscillating cylinder. For all of these cases, the present method shows better accuracy in terms of force coefficients or representative parameters in comparison to the IBM in FOAM-Extend 4.0 with the same mesh. In addition, there is neither non-physical penetration of the streamlines nor non-physical oscillations in the force coefficients or pressure distribution. In the end, the EIB-LBFS method is further validated by solving an FSI problem. These tests provide evident proofs that the EIB-LBFS method is reliable and that it could be a promising alternative for solving incompressible flow problems with a moving boundary and fluid–structure interaction problems.

Author Contributions: Conceptualization, Y.L. (Yangyang Liu), Z.Z. and Y.L. (Yaguang Liu); methodology, Y.L. (Yangyang Liu), Y.L. (Yaguang Liu) and H.Z.; software, Y.L. (Yaguang Liu) and H.Z.; validation, Y.L. (Yangyang Liu), Y.L. (Yaguang Liu) and H.Z.; investigation, Z.Z.; writing—original draft preparation, Y.L. (Yangyang Liu) and Y.L. (Yaguang Liu); writing—review and editing, Y.L. (Yangyang Liu), Z.Z., H.Z. and Y.L. (Yaguang Liu); funding acquisition, Y.L. (Yangyang Liu) and Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Natural Science Foundation of Chongqing, grant number cstc2021jcyj-msxmX0010, and the Science and Technology Research Program of Chongqing Municipal Education Commission, grant number KJQN202101117.

Data Availability Statement: All data generated are shown in the text and illustrations.

Conflicts of Interest: The authors declare no conflicts of interest.

Nomenclature

ρ	density
$\boldsymbol{u}$	velocity vector
p	pressure
μ	dynamic viscosity
ν	kinematic viscosity
$\boldsymbol{f}$	restoring force
$\boldsymbol{U}$	conservative variables vector
$\boldsymbol{F}$	flux tensor
$\boldsymbol{G}$	body force vector
Ω_i	control cell i
$\boldsymbol{e}_\alpha$	lattice velocity vector
τ_ν	single relaxation parameter
δ_t	streaming time step
h	mesh spacing
c_s	the sound speed
f^{eq}	equilibrium density distribution function
f^{neq}	non-equilibrium density distribution function
$\boldsymbol{r}$	physical location of the cell interface
t	time
$\Delta\boldsymbol{u}$	velocity correction
$\boldsymbol{X}_l$	Lagrangian points
$\Delta\boldsymbol{u}_B(\boldsymbol{X}_l)$	velocity corrections at the Lagrangian points
D_c	diameter of the circular cylinder
Re	Reynolds number
U_0	free-stream velocity
C_p	pressure coefficient

C_l	lift coefficient
C_d	drag coefficient
St	Strouhal number
L_s	recirculation length
θ_s	separation angle
p_0	pressure of the free-stream
p_w	pressure on the cylinder surface
F_l	lift force
F_d	drag force
f_o	vortex shedding frequency
A_e	oscillating amplitude
f_e	excitation frequency

References

1. Sotiropoulos, F.; Yang, X. Immersed boundary methods for simulating fluid-structure interaction. *Prog. Aerosp. Sci.* **2014**, *65*, 1–21. [CrossRef]
2. Wu, B.; Shu, C.; Wan, M.; Wang, Y.; Chen, S. Hydrodynamic performance of an unconstrained flapping swimmer with flexible fin: A numerical study. *Phys. Fluids* **2022**, *34*, 011901. [CrossRef]
3. Zabala, I.; Henriques, J.C.C.; Blanco, J.M.; Gomez, A.; Gato, L.M.C.; Bidaguren, I.; Falcão, A.F.O.; Amezaga, A.; Gomes, R.P.F. Wave-induced real-fluid effects in marine energy converters: Review and application to OWC devices. *Renew. Sustain. Energy Rev.* **2019**, *111*, 535–549. [CrossRef]
4. Zhu, C.; Yu, Z.; Shao, X. Interface-resolved direct numerical simulations of the interactions between neutrally buoyant spheroidal particles and turbulent channel flows. *Phys. Fluids* **2018**, *30*, 115103. [CrossRef]
5. Zhu, C.; Yu, Z.; Pan, D.; Shao, X. Interface-resolved direct numerical simulations of the interactions between spheroidal particles and upward vertical turbulent channel flows. *J. Fluid Mech.* **2020**, *891*, A6. [CrossRef]
6. Zhang, Z.L.; Shu, C.; Liu, Y.Y.; Liu, W.; Khalid, M.S.U. An improved M-SPEM for modeling complex hydroelastic fluid-structure interaction problems. *J. Comput. Phys.* **2023**, *488*, 112233. [CrossRef]
7. Peskin, C.S. The immersed boundary method. *Acta Numer.* **2002**, *11*, 479–517. [CrossRef]
8. Mittal, R.; Iaccarino, G. Immersed boundary method. *Annu. Rev. Fluid Mech.* **2005**, *37*, 239–261. [CrossRef]
9. Huang, W.X.; Shin, S.J.; Sung, H.J. Simulation of flexible filaments in a uniform flow by the immersed boundary method. *J. Comput. Phys.* **2007**, *226*, 2206–2228. [CrossRef]
10. Wu, J.; Shu, C. Implicit velocity correction-based immersed boundary-lattice Boltzmann method and its applications. *J. Comput. Phys.* **2009**, *228*, 1963–1979. [CrossRef]
11. Wu, J.; Shu, C. An improved immersed boundary-lattice Boltzmann method for simulating three-dimensional incompressible flows. *J. Comput. Phys.* **2010**, *229*, 5022–5042. [CrossRef]
12. Peskin, C.S. Flow patterns around heart valves: A numerical method. *J. Comput. Phys.* **1972**, *10*, 252–271. [CrossRef]
13. Lai, M.C.; Peskin, C.S. An immersed boundary method with formal second-order accuracy and reduced numerical viscosity. *J. Comput. Phys.* **2000**, *160*, 705–719. [CrossRef]
14. Kim, Y.; Peskin, C.S. Penalty immersed boundary method for an elastic boundary with mass. *Phys. Fluids* **2007**, *19*, 053103. [CrossRef]
15. Huang, W.X.; Tian, F.B. Recent trends and progress in the immersed boundary method. *Proc. Inst. Mech. Eng. Part C J. Mech. Eng. Sci.* **2019**, *233*, 7617–7636. [CrossRef]
16. Goldstein, D.; Handler, R.; Sirovich, L. Modeling a no-slip flow boundary with an external force field. *J. Comput. Phys.* **1993**, *105*, 354–366. [CrossRef]
17. Feng, Z.G.; Michaelides, E.E. Proteus: A direct forcing method in the simulations of particulate flows. *J. Comput. Phys.* **2005**, *202*, 20–51. [CrossRef]
18. Park, H.; Pan, X.; Lee, C.; Choi, J.-I. A pre-conditioned implicit direct forcing based immersed boundary method for incompressible viscous flows. *J. Comput. Phys.* **2016**, *314*, 774–799. [CrossRef]
19. Uhlmann, M. An immersed boundary method with direct forcing for the simulation of particulate flows. *J. Comput. Phys.* **2005**, *209*, 448–476. [CrossRef]
20. Niu, X.; Shu, C.; Chew, Y.; Peng, Y. A momentum exchange-based immersed boundary-lattice Boltzmann method for simulating incompressible viscous flows. *Phys. Lett. A* **2006**, *354*, 173–182. [CrossRef]
21. Jasak, H.; Rigler, D.; Tuković, Ž. Design and implementation of immersed boundary method with discrete forcing approach for boundary conditions. In Proceedings of the 11th World Congress on Computational Mechanics, WCCM 2014, 5th European Conference on Computational Mechanics, ECCM 2014 and 6th European Conference on Computational Fluid Dynamics, ECFD 2014, Barcelona, Spain, 20–25 July 2014; International Center for Numerical Methods in Engineering: Barcelona, Spain, 2014; pp. 5319–5332.
22. Giahi, M.; Bergstrom, D. A critical assessment of the immersed boundary method for modeling flow around fixed and moving bodies. *Comput. Fluids* **2023**, *256*, 105841. [CrossRef]

23. Mohd-Yusof, J. Interaction of Massive Particles with Turbulence. Ph.D. Thesis, Cornell University, Ithaca, NY, USA, 1996.
24. Fadlun, E.; Verzicco, R.; Orlandi, P.; Mohd-Yusof, J. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations. *J. Comput. Phys.* **2000**, *161*, 35–60. [CrossRef]
25. Zhang, N.; Zheng, Z. An improved direct-forcing immersed-boundary method for finite difference applications. *J. Comput. Phys.* **2007**, *221*, 250–268. [CrossRef]
26. Azis, M.H.A.; Evrard, F.; van Wachem, B. An immersed boundary method for incompressible flows in complex domains. *J. Comput. Phys.* **2019**, *378*, 770–795. [CrossRef]
27. Li, Z.; Cao, W.; Le Touzé, D. On the coupling of a direct-forcing immersed boundary method and the regularized lattice Boltzmann method for fluid-structure interaction. *Comput. Fluids* **2019**, *190*, 470–484. [CrossRef]
28. Luo, K.; Wang, Z.; Fan, J.; Cen, K. Full-scale solutions to particle-laden flows: Multidirect forcing and immersed boundary method. *Phys. Rev. E* **2007**, *76*, 066709. [CrossRef]
29. Wang, Z.; Fan, J.; Luo, K. Combined multi-direct forcing and immersed boundary method for simulating flows with moving particles. *Int. J. Multiph. Flow* **2008**, *34*, 283–302. [CrossRef]
30. Zhao, X.; Chen, Z.; Yang, L.; Liu, N.; Shu, C. Efficient boundary condition-enforced immersed boundary method for incompressible flows with moving boundaries. *J. Comput. Phys.* **2021**, *441*, 110425. [CrossRef]
31. Pinelli, A.; Naqavi, I.; Piomelli, U.; Favier, J. Immersed-boundary methods for general finite-difference and finite-volume Navier-Stokes solvers. *J. Comput. Phys.* **2010**, *229*, 9073–9091. [CrossRef]
32. Li, Z.; Favier, J.; D'Ortona, U.; Poncet, S. An immersed boundary-lattice Boltzmann method for single- and multi-component fluid flows. *J. Comput. Phys.* **2016**, *304*, 424–440. [CrossRef]
33. Zhang, D.; Li, S.; Jiao, S.; Shang, Y.; Dong, M. Relative permeability of three immiscible fluids in random porous media determined by the lattice Boltzmann method. *Int. J. Heat Mass Transf.* **2019**, *134*, 311–320. [CrossRef]
34. Tian, F.B.; Luo, H.; Zhu, L.; Liao, J.C.; Lu, X.Y. An efficient immersed boundary-lattice Boltzmann method for the hydrodynamic interaction of elastic filaments. *J. Comput. Phys.* **2011**, *230*, 7266–7283. [CrossRef]
35. Zhang, D.; Li, Y.; Zhang, J.; Yuan, H.; Zhang, Z.L. Pore-scale numerical study: Brine water crystallization with ice crystal particle motion using the LBM-PFM-IBM. Appl. *Therm. Eng.* **2023**, *234*, 121258. [CrossRef]
36. Li, Q.Z.; Lu, Z.L.; Chen, Z.; Shu, C.; Liu, Y.Y.; Guo, T.Q. A simplified lattice Boltzmann model for two-phase electro-hydrodynamics flows and its application to simulations of droplet deformation in electric field. *Appl. Math. Model.* **2023**, *122*, 99–126. [CrossRef]
37. Wang, Y.; Shu, C.; Teo, C.; Wu, J. An immersed boundary-lattice Boltzmann flux solver and its applications to fluid-structure interaction problems. *J. Fluids Struct.* **2015**, *54*, 440–465. [CrossRef]
38. Wang, Y.; Shu, C.; Yang, L. Boundary condition-enforced immersed boundary-lattice Boltzmann flux solver for thermal flows with Neumann boundary conditions. *J. Comput. Phys.* **2016**, *306*, 237–252. [CrossRef]
39. Shu, C.; Wang, Y.; Teo, C.J.; Wu, J. Development of lattice Boltzmann flux solver for simulation of incompressible flows. *Adv. Appl. Math. Mech.* **2014**, *6*, 436–460. [CrossRef]
40. Liu, Y.Y.; Shu, C.; Zhang, H.W.; Yang, L.M. A high order least square-based finite difference-finite volume method with lattice Boltzmann flux solver for simulation of incompressible flows on unstructured grids. *J. Comput. Phys.* **2020**, *401*, 109019. [CrossRef]
41. Liu, Y.Y.; Yang, L.M.; Shu, C.; Zhang, H.W. Three-dimensional high-order least square-based finite difference-finite volume method on unstructured grids. *Phys. Fluids* **2020**, *32*, 123604. [CrossRef]
42. Zhang, H.; Liu, Y.; Zhang, Z.; Wang, L.P.; Shu, C. An immersed boundary-lattice Boltzmann flux solver for simulation of flows around structures with large deformation. *Phys. Fluids* **2023**, *35*, 031912. [CrossRef]
43. Liu, Y.Y.; Yang, L.M.; Shu, C.; Zhang, H.W. Efficient high-order radial basis-function-based differential quadrature-finite volume method for incompressible flows on unstructured grids. *Phys. Rev. E* **2021**, *104*, 045312. [CrossRef]
44. Liu, Y.Y.; Shu, C.; Zhang, H.W.; Yang, L.M. A high-order implicit least square-based finite difference-finite volume method for incompressible flows on unstructured grids. *Phys. Fluids* **2021**, *33*, 053601. [CrossRef]
45. Liu, Y.G.; Shu, C.; Yu, P.; Liu, Y.Y.; Zhang, H.; Lu, C. A high-order generalized differential quadrature method with lattice Boltzmann flux solver for simulating incompressible flows. *Phys. Fluids* **2023**, *35*, 047107.
46. Liu, Y.Y.; Yang, L.M.; Shu, C.; Zhang, Z.L.; Yuan, Z.Y. An implicit high-order radial basis function-based differential quadrature-finite volume method on unstructured grids to simulate incompressible flows with heat transfer. *J. Comput. Phys.* **2022**, *467*, 111461. [CrossRef]
47. Wu, B.; Lu, J.; Lee, H.C.; Shu, C.; Wan, M. An explicit boundary condition-enforced immersed boundary-reconstructed thermal lattice Boltzmann flux solver for thermal-fluid-structure interaction problems with heat flux boundary conditions. *J. Comput. Phys.* **2023**, *485*, 112106. [CrossRef]
48. Yang, L.M.; Shu, C.; Wu, J. A moment conservation-based non-free parameter compressible lattice Boltzmann model and its application for flux evaluation at cell interface. *Comput. Fluids* **2013**, *79*, 190–199. [CrossRef]
49. Ma, C.; Wu, J.; Yu, H.; Yang, L. A high-order implicit-explicit flux reconstruction lattice Boltzmann method for viscous incompressible flows. *Comput. Math. Appl.* **2022**, *105*, 13–28. [CrossRef]
50. Dennis, S.C.R.; Chang, G.Z. Numerical solutions for steady flow past a circular cylinder at Reynolds numbers up to 100. *J. Fluid Mech.* **1970**, *42*, 471–489. [CrossRef]
51. Shukla, R.K.; Tatineni, M.; Zhong, X. Very high-order compact finite difference schemes on non-uniform grids for incompressible Navier-Stokes equations. *J. Comput. Phys.* **2007**, *224*, 1064–1094. [CrossRef]

52. He, X.; Doolen, G. Lattice Boltzmann method on curvilinear coordinates system: Flow around a circular cylinder. *J. Comput. Phys.* **1997**, *134*, 306–315. [CrossRef]
53. Pellerin, N.; Leclaire, S.; Reggio, M. Solving incompressible fluid flows on unstructured meshes with the lattice Boltzmann flux solver. *Eng. Appl. Comput. Fluid Mech.* **2017**, *11*, 310–327. [CrossRef]
54. Braza, M.; Chassaing, P.; Minh, H.H. Numerical study and physical analysis of the pressure and velocity fields in the near wake of a circular cylinder. *J. Fluid Mech.* **1986**, *165*, 79–130. [CrossRef]
55. Liu, C.; Zheng, X.; Sung, C.H. Preconditioned multigrid methods for unsteady incompressible flows. *J. Comput. Phys.* **1998**, *139*, 35–57. [CrossRef]
56. Posdziech, O.; Grundmann, R. A systematic approach to the numerical calculation of fundamental quantities of the two-dimensional flow over a circular cylinder. *J. Fluids Struct.* **2007**, *23*, 479–499. [CrossRef]
57. Persillon, H.; Braza, M. Physical analysis of the transition to turbulence in the wake of a circular cylinder by three-dimensional Navier-Stokes simulation. *J. Fluid Mech.* **1998**, *365*, 23–88. [CrossRef]
58. Franke, R.; Rodi, W.; Schönung, B. Numerical calculation of laminar vortex-shedding flow past cylinders. *J. Wind. Eng. Ind. Aerodyn.* **1990**, *35*, 237–257. [CrossRef]
59. Imamura, T.; Suzuki, K.; Nakamura, T.; Yoshida, M. Flow simulation around an airfoil by lattice Boltzmann method on generalized coordinates. *AIAA J.* **2005**, *43*, 1968–1973. [CrossRef]
60. Guilmineau, E.; Queutey, P. A numerical simulation of vortex shedding from an oscillating circular cylinder. *J. Fluids Struct.* **2002**, *16*, 773–794. [CrossRef]
61. Feng, Z.G.; Michaelides, E.E. The immersed boundary-lattice Boltzmann method for solving fluid-particles interaction problems. *J. Comput. Phys.* **2004**, *195*, 602–628. [CrossRef]
62. Wan, D.; Turek, S. Direct numerical simulation of particulate flow via multigrid FEM techniques and the fictitious boundary method. *Int. J. Numer. Methods Fluids* **2006**, *51*, 531–566. [CrossRef]