Journal of Computational Physics 530 (2025) 113913

Contents lists available at ScienceDirect

Journal of Computational Physics

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

An implicit moving-least-squares immersed boundary method for high-fidelity fluid-structure interaction simulations

Buchen Wu [a], Lin Fu [a,b,*]

[a] Department of Mechanical and Aerospace Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

[b] Department of Mathematics, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

ARTICLE INFO

Keywords:
Fluid-structure interaction
Moving-least-squares
Immersed boundary method
Moving-boundary problems
Low-speed flows

ABSTRACT

In this work, an implicit moving-least-squares immersed boundary method (MLS-IBM) is proposed to accurately enforce the velocity boundary condition on immersed objects. This method effectively eliminates errors induced by the inequality between the interpolation and spreading operations, while preserving the conservation of the force and torque. The instantaneous discretization errors for the velocity boundary conditions are negligible, approaching machine round-off levels, which renders the proposed method much more accurate than previous MLS-IBMs. In terms of computational efficiency, the proposed implicit MLS-IBM outperforms the explicit variant MLS-IBM for stationary problems and shows comparable performance for moving-boundary problems. Additionally, the assembly of the correlation matrix in the implicit MLS-IBM is optimized to improve the computational efficiency, making it superior to previous implicit IBMs. The proposed implicit MLS-IBM integrated with the lattice Boltzmann flux solver can achieve second-order spatial accuracy through a mesh-refinement study. The robustness and accuracy of the proposed implicit MLS-IBM are validated through several complex fluid-structure interaction (FSI) problems involving complex geometries, moving boundaries, and large deformations.

1. Introduction

Fluid-structure interaction (FSI) phenomena have garnered significant attention across various engineering areas, such as wind engineering [1], ocean engineering [2] and chemical engineering [3]. Surrounding flows can induce complex nonlinear vibrations and large deformations in immersed objects, making the interaction between solid and fluid phases a challenge for computational fluid dynamics (CFD). To achieve high numerical accuracy on the immersed boundary, conventional numerical methods adopt body-fitted meshes to solve FSI problems. However, for FSI problems involving complex geometries and moving boundaries, providing high-quality body-fitted meshes at each time step requires substantial computational efforts consumed by the advanced mesh generation technique. As an alternative to conventional numerical methods using body-fitted meshes, the immersed boundary method (IBM) incorporates a source term into the momentum equation or directly modifies the local solutions to mimic the solid boundary effects on non-conformal Cartesian meshes. Consequently, IBM eliminates the need for mesh regeneration when dealing with geometry deformations and moving boundaries.

* Corresponding author at: Department of Mechanical and Aerospace Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong.
E-mail address: linfu@ust.hk (L. Fu).

https://doi.org/10.1016/j.jcp.2025.113913
Received 20 September 2024; Received in revised form 10 January 2025; Accepted 4 March 2025

Available online 6 March 2025

The concept of IBM is firstly proposed by Peskin [4] in 1970s, where the IBM is employed to simulate how the blood flows interact with heart valves. The IBM developed by Peskin [4] is a penalty forcing method, in which massive and massless material points are introduced to interact with solid structures and surrounding flows, respectively. The restoring forces are calculated by constraining the massive and massless material points to maintain close proximity through Hooke's law with an artificial spring constant. Inspired by the penalty forcing IBM, a variety of IBM variants have been developed, utilizing different physical models to calculate the restoring forces. To remove the artificial spring constant in the penalty forcing IBM, Fadlun et al. [5] proposed a direct forcing IBM, where the restoring force is designed to eliminate the difference between the reconstructed velocity on the immersed boundary and the desired velocity boundary condition. Feng and Michaelides [6] incorporate the direct forcing IBM into the lattice Boltzmann method (LBM) to simulate particle-laden flows. To alleviate the strong oscillations present in the original direct forcing IBM, Uhlmann [7] introduces the regularized delta functions into the direct forcing IBM, which simplifies the interpolation and spreading processes and preserves the conservation of force and torque. Consequently, this direct forcing IBM has been employed to solve various FSI problems, such as vortex-induced vibrations [8] and particle-laden flows [9,10]. However, the volume restoring forces predicted by this one-step explicit direct forcing IBM are inconsistent with the desired values for satisfying the boundary condition, due to the inequality between the interpolation and spreading operations [11,12]. There are two approaches to accurately predict the restoring forces, namely, iterative method and implicit method. Luo et al. [13] propose the multi-direct forcing IBM, which iteratively implements the direct forcing IBM to ensure that velocity errors satisfy the convergence criteria. However, for FSI problems with multiple moving objects, the iteration process consumes significant computational effort. Additionally, while the multi-direct forcing IBM preserves the conservation in each internal iteration, the total force and torque produced across all internal iterations are not conserved [14]. Wu and Shu [15] first propose the implicit velocity correction IBM, where the errors induced by the inequality between the interpolation and spreading processes are eliminated; consequently, the boundary velocity errors are reduced to a level negligible compared to machine round-off. Subsequently, several implicit direct forcing IBMs have been proposed [11,16], which are equivalent to the implicit velocity correction IBM [17].

The aforementioned IBMs commonly adopt the regularized delta functions for the interpolation and spreading operations; however, the form of the delta function significantly affects the predicted results [18,19]. To eliminate the artificial effects of the regularized delta functions and preserve accuracy on the immersed boundary, Vanella and Balaras [20] replace the delta function with the moving-least-square (MLS) approximation in the direct forcing IBM, namely the MLS-IBM. Subsequently, the MLS-IBM has been extended to simulate FSI problems with elastic objects [21,22] and has been coupled with LBM to compare the performance of "*dynamic*" and "*kinematic*" IBMs [23]. However, the original MLS-IBM also contains the errors induced by the inequality between the interpolation and spreading operations. Recently, Chen et al. [14] propose an explicit variant MLS-IBM to reduce the errors in the velocity boundary condition by introducing a correction coefficient in the force spreading process, where the correction coefficient is determined by the least-squares method. However, the numerical results presented in Chen et al. [14] demonstrate that the errors in the velocity boundary condition remain noticeable, indicating that the no-slip boundary condition is not accurately enforced, and this explicit technique cannot be extended to high-order numerical frameworks. Meanwhile, Chen et al. [14] also clarify that the correction coefficient employed in the spreading process breaks the conservation of force and torque in the MLS-IBM, which is a fundamental and crucial property of IBM. In addition, for both stationary and moving problems, the correction coefficient must be calculated at each time step through a complex algebraic operation.

In this work, to simultaneously eliminate the boundary velocity discretization errors and preserve the conservation of force and torque, we propose an implicit MLS-IBM, where the errors induced by the inequality between the interpolation and spreading operations are effectively eliminated. The restoring forces are implicitly evaluated from the differences between the desired boundary velocity and the intermediate reconstructed flow velocity, which can accurately enforce the no-slip boundary condition on the immersed boundary. Moreover, we optimize the intrinsic disadvantages in the assembly of the correlation matrix in the traditional implicit IBMs to significantly improve the computational efficiency. The rest of the paper is organized as follows. Section 2 presents the governing equations, the numerical method to decompose the governing equations, and the approach for simulating incompressible viscous flows. In Section 3, the methodologies of the original MLS-IBM and the explicit variant MLS-IBM are described. Section 4 introduces the methodology of the proposed implicit MLS-IBM, along with the comparisons of the boundary velocity errors, errors in the conservation of force and torque, and computational efficiency between the MLS-IBM, the explicit variant MLS-IBM, the traditional implicit IBMs and the proposed implicit MLS-IBM. In Section 5, the global accuracy test and numerical validations of the proposed implicit MLS-IBM are conducted. Conclusions are provided in Section 6.

2. The governing equations and numerical methods

In this section, the numerical approach employed to solve the governing equations of fluid-structure interaction problems is presented. In the present work, the solution process of Navier-Stokes equations is decomposed into prediction and forcing steps. In the prediction step, the lattice Boltzmann flux solver (LBFS) [24,25] is adopted to predict the intermediate flow field, where the solid boundary effects are excluded. Subsequently, the IBM is applied to evaluate restoring force terms to correct the intermediate flow field, ensuring that the no-slip boundary condition is satisfied in the forcing step.

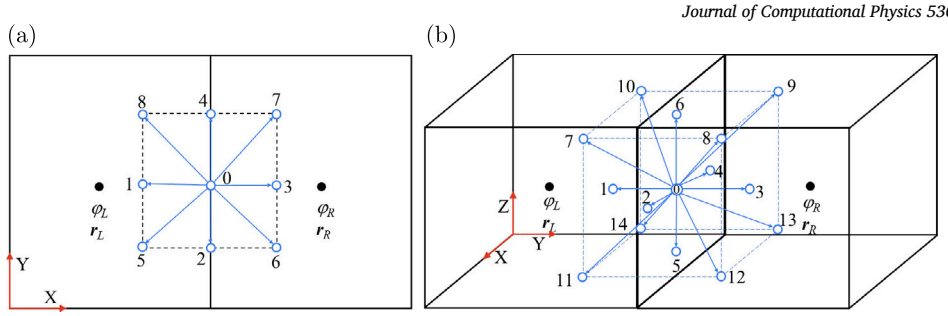

Fig. 1. Local reconstruction of LBM solution with (a) D2Q9 model for two-dimensional space and (b) D3Q15 model for three-dimensional space at cell interface between two adjacent cells in the LBFS.

2.1. Navier-Stokes (N-S) equations

The flow dynamics can be described by the mass and momentum conservation laws, which can be written as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{P} = 0, \tag{1a}$$

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

$$\mathbf{P} = \rho \mathbf{u}, \quad \mathbf{\Pi} = \rho \mathbf{u}\mathbf{u} + p\boldsymbol{I} - \mu \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^{\mathrm{T}} \right), \tag{1c}$$

where ρ, $\mathbf{u}$, p, $\boldsymbol{I}$ and μ represent the fluid density, flow velocity, pressure, unit tensor and dynamic viscosity, respectively. $\mathbf{f}$ denotes the restoring force term, which can be calculated by the IBM. The above N-S equations can be employed to simulate incompressible flows with a low Mach number, where the density variation is relatively small. In the present work, the N-S equations are solved by a fractional step method, where the whole solution process can be divided into prediction and forcing steps. In the prediction step, the solid boundary effects are excluded, so the restoring forcing term is set to $\mathbf{f} = 0$ in Eq. (1b). In the forcing step, the restoring forcing term can be evaluated based on the difference between the desired boundary condition and the intermediate flow variables around the immersed boundary. Subsequently, the restoring forcing term evaluated by the IBM is employed to correct the intermediate flow field for accurately enforcing the physical boundary condition. The whole solution process can be written as

$$\begin{cases} \frac{\rho^{n+1} - \rho^n}{\Delta t} + \nabla \cdot \mathbf{P} = 0, \\ \frac{\rho^{n+1}\mathbf{u}^* - \rho^n \mathbf{u}^n}{\Delta t} + \nabla \cdot \mathbf{\Pi} = 0, \end{cases} \quad \text{prediction step;} \tag{2a}$$

$$\frac{\rho^{n+1}\mathbf{u}^{n+1} - \rho^{n+1}\mathbf{u}^*}{\Delta t} = \mathbf{f}, \quad \text{forcing step.} \tag{2b}$$

2.2. The intermediate flow variables predicted by the LBFS

In this subsection, the numerical approach used to solve the governing equations in the prediction step is described. In the present work, the governing equations are discretized by a cell-centered finite volume method, where the LBFS is adopted to evaluate the viscous and inviscid fluxes at all cell interfaces of each control volume simultaneously. The numerical fluxes in the governing equations can be evaluated by the local distribution function though the following relationship as

$$\mathbf{P} = \rho \mathbf{u} = \sum_{\alpha=0}^{N} f_\alpha^{\mathrm{eq}} \boldsymbol{e}_\alpha, \tag{3a}$$

$$\mathbf{\Pi} = \rho \mathbf{u}\mathbf{u} + p\boldsymbol{I} - \mu \left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T \right) = \sum_{\alpha=0}^{N} \boldsymbol{e}_\alpha \boldsymbol{e}_\alpha \hat{f}_\alpha, \tag{3b}$$

$$\hat{f}_\alpha = f_\alpha^{\mathrm{eq}} + \left(1 - \frac{1}{2\tau} \right) f_\alpha^{\mathrm{neq}}, \tag{3c}$$

$$f_\alpha^{\mathrm{neq}} = -\tau \delta t \left(\frac{\partial}{\partial t} + \boldsymbol{e}_\alpha \cdot \nabla \right) f_\alpha^{\mathrm{eq}}, \tag{3d}$$

where f_α^{eq} and f_α^{neq} represent the equilibrium and nonequilibrium density distribution functions, respectively. δt, $\boldsymbol{e}_\alpha$ and τ denote the streaming time step, the particle velocity in the α direction and the single relaxation parameter, respectively. Therefore, the key step in evaluating the numerical fluxes is to accurately determine f_α^{eq} and f_α^{neq}.

In the present work, the D2Q9 and D3Q15 lattice velocity models (see Fig. 1) are employed for simulating 2D and 3D problems, respectively. According to Eq. (3d), f_α^{neq} only depends on the f_α^{eq} at the cell interface and its derivatives. Through employing the second-order Taylor series expansion, f_α^{neq} can be simplified as

$$f_\alpha^{\mathrm{neq}}(\boldsymbol{r},t) = -\tau[f_\alpha^{\mathrm{eq}}(\boldsymbol{r},t) - f_\alpha^{\mathrm{eq}}(\boldsymbol{r}-\boldsymbol{e}_\alpha\delta t, t-\delta t)], \tag{4}$$

where $\boldsymbol{r}$ and $\boldsymbol{r}-\boldsymbol{e}_\alpha\delta t$ represent the positions of the 0-th point and the surrounding lattice points, respectively. Particles at surrounding lattice points are streamed to the 0-th point at the cell interface during the streaming time interval δt. Within the framework of the LBM, the equilibrium density distribution function $f_\alpha^{\mathrm{eq}}(\boldsymbol{r}-\boldsymbol{e}_\alpha\delta t, t-\delta t)$ can be recovered from local flow density $\rho(\boldsymbol{r}-\boldsymbol{e}_\alpha\delta t, t-\delta t)$ and velocity $\mathbf{u}(\boldsymbol{r}-\boldsymbol{e}_\alpha\delta t, t-\delta t)$ at position $\boldsymbol{r}-\boldsymbol{e}_\alpha\delta t$ and time $t-\delta t$ by using the LBGK model as

$$f_\alpha^{eq}(\boldsymbol{r},t) = \rho w_\alpha\left[1 + \frac{\boldsymbol{e}_\alpha\cdot\mathbf{u}}{c_s^2} + \frac{\left(\boldsymbol{e}_\alpha\cdot\mathbf{u}\right)^2 - \left(c_s|\mathbf{u}|\right)^2}{2c_s^4}\right], \tag{5}$$

where w_α and c_s denote the coefficient in the lattice velocity model and the sound speed, respectively. Based on the literature [26], the w_α is given as

$$\text{D2Q9 model: } w_\alpha = \begin{cases} \frac{4}{9}, & \alpha = 0, \\ \frac{1}{9}, & \alpha = 1-4, \\ \frac{1}{36}, & \alpha = 5-8; \end{cases} \quad \text{D3Q15 model: } w_\alpha = \begin{cases} \frac{2}{9}, & \alpha = 0, \\ \frac{1}{9}, & \alpha = 1-6, \\ \frac{1}{72}, & \alpha = 7-14. \end{cases} \tag{6}$$

c_s is defined as $c_s = c/\sqrt{3}$, where c is set to 1. The density and flow velocity at lattice points can be obtained through the following interpolation formulation as

$$\phi(\boldsymbol{r}-\boldsymbol{e}_\alpha\delta t, t-\delta t) = \begin{cases} \phi_L + \nabla\phi_L\cdot\left(\boldsymbol{r}-\boldsymbol{e}_\alpha\delta t-\boldsymbol{r}_L\right), \boldsymbol{r}-\boldsymbol{e}_\alpha\delta t \text{ locates at the left cell;} \\ \phi_R + \nabla\phi_R\cdot\left(\boldsymbol{r}-\boldsymbol{e}_\alpha\delta t-\boldsymbol{r}_R\right), \boldsymbol{r}-\boldsymbol{e}_\alpha\delta t \text{ locates at the right cell;} \end{cases} \tag{7}$$

where ϕ_L and ϕ_R represent the macroscopic flow variables at two adjacent cell centers $\boldsymbol{r}_L$ and $\boldsymbol{r}_R$, respectively. Through applying the LBGK model, $f_\alpha^{\mathrm{eq}}(\boldsymbol{r},t)$ can be determined with the flow variables $\rho(\boldsymbol{r},t)$ and $\mathbf{u}(\boldsymbol{r},t)$. The macroscopic flow variables at the cell interface can be reconstructed by the LBM solutions as

$$\rho(\boldsymbol{r},t) = \sum_{\alpha=0}^{N} f_\alpha^{\mathrm{eq}}(\boldsymbol{r}-\boldsymbol{e}_\alpha\delta t, t-\delta t), \tag{8a}$$

$$\rho(\boldsymbol{r},t)\mathbf{u}(\boldsymbol{r},t) = \sum_{\alpha=0}^{N} f_\alpha^{\mathrm{eq}}(\boldsymbol{r}-\boldsymbol{e}_\alpha\delta t, t-\delta t)\boldsymbol{e}_\alpha. \tag{8b}$$

Subsequently, following Eq. (4), the obtained $f_\alpha^{\mathrm{eq}}(\boldsymbol{r},t)$ and $f_\alpha^{\mathrm{eq}}(\boldsymbol{r}-\boldsymbol{e}_\alpha\delta t, t-\delta t)$ are applied to calculate $f_\alpha^{\mathrm{neq}}(\boldsymbol{r},t)$. Once f_α^{eq} and f_α^{neq} are obtained, the numerical fluxes $\mathbf{P}$ and $\mathbf{\Pi}$ can be calculated though Eq. (3).

Consequently, with the obtained numerical fluxes at cell interfaces, the flow variables ρ^{n+1} and $\mathbf{u}^*$ at the cell center can be updated through

$$\frac{\rho^{n+1}-\rho^n}{\Delta t} = -\frac{1}{\Delta V}\sum_k \mathbf{P}_k\Delta S_k, \tag{9a}$$

$$\frac{\rho^{n+1}\mathbf{u}^*-\rho^n\mathbf{u}^n}{\Delta t} = -\frac{1}{\Delta V}\sum_k \mathbf{\Pi}_k\Delta S_k, \tag{9b}$$

where the superscripts "n", "$*$" and "$n+1$" denote the current, intermediate and next time steps, respectively. Δt, ΔV and ΔS are the macroscopic time interval, the control cell volume and the area of interface, respectively. The LBFS can achieve second-order spatial accuracy [25]. In this work, the third-order strong-stability-preserving (SSP) Runge-Kutta method is employed for the time discretization. The numerical approach for solving fluid dynamics is not the focus of the present work, readers are referred to the previous works [24,25] for detailed information on the LBFS.

3. Existing moving-least-squares immersed boundary methods

In this section, the methodology of the original MLS-IBM and the explicit variant MLS-IBM are presented.

3.1. The original MLS-IBM

See Fig. 2, in the framework of the original MLS-IBM [20], the flow variables at each Lagrangian point can be reconstructed by using the MLS method, which can be expressed as

$$\mathbf{U}^{*,i}\left(\mathbf{X}^i\right) = \boldsymbol{p}^{\mathrm{T}}\left(\mathbf{X}^i\right)\boldsymbol{a}\left(\mathbf{X}^i\right) = \sum_{j=1}^{m} p_j\left(\mathbf{X}^i\right)a_j\left(\mathbf{X}^i\right), \tag{10}$$

where $\mathbf{X}^i$ denotes the position of the i-th Lagrangian point. $\boldsymbol{p}\left(\mathbf{X}^i\right)$ denotes the basis function vector with dimension m. $\boldsymbol{a}\left(\mathbf{X}^i\right)$ is the vector of coefficients. In the present work, a linear basis function $\boldsymbol{p}^{\mathrm{T}}\left(\mathbf{X}^i\right) = \left[1, X^i, Y^i, Z^i\right]$ is employed, which can achieve high

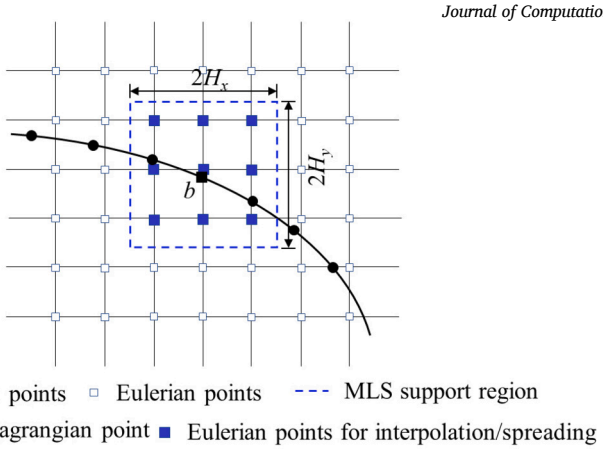

Fig. 2. Illustration of the original MLS-IBM, where b denotes the selected Lagrangian point, $2H_x$ and $2H_y$ represent the size of the MLS support region in the x and y directions, respectively. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

computational efficiency and preserve the accuracy of the reconstructed flow variables up to the spatial discretization scheme. The vector of coefficients $\boldsymbol{a}\left(\mathbf{X}^i\right)$ can be determined by minimizing the weighted L2-norm, which can be expressed as

$$J=\sum_{j=1}^{N_e} W\left(\mathbf{X}^i-\mathbf{x}^j\right)\left[\boldsymbol{p}^{\mathrm{T}}\left(\mathbf{x}^j\right)\boldsymbol{a}\left(\mathbf{X}^i\right)-u^{*,j}\right], \tag{11}$$

where $W\left(\mathbf{X}^i-\mathbf{x}^j\right)$ denotes the given weight function, $\mathbf{x}^j$ represents the physical position of the j-th Eulerian point within the interpolation region, and $u^{*,j}$ is the intermediate velocity component. N_e represents the number of the Eulerian points affected by the i-th Lagrangian point. The minimizing operation can give the following relationships:

$$\boldsymbol{A}\left(\mathbf{X}^i\right)\boldsymbol{a}\left(\mathbf{X}^i\right)=\boldsymbol{B}\left(\mathbf{X}^i\right)\boldsymbol{u}^*, \tag{12a}$$

$$\boldsymbol{A}\left(\mathbf{X}^i\right)=\sum_{j=1}^{N_e} W\left(\mathbf{X}^i-\mathbf{x}^j\right)\boldsymbol{p}\left(\mathbf{x}^j\right)\boldsymbol{p}^{\mathrm{T}}\left(\mathbf{x}^j\right), \tag{12b}$$

$$\boldsymbol{B}\left(\mathbf{X}^i\right)=\left[W\left(\mathbf{X}^i-\mathbf{x}^1\right)\boldsymbol{p}\left(\mathbf{x}^1\right)\cdots W\left(\mathbf{X}^i-\mathbf{x}^{N_e}\right)\boldsymbol{p}\left(\mathbf{x}^{N_e}\right)\right], \tag{12c}$$

$$\boldsymbol{u}^*=\left[u^{*,1}\cdots u^{*,N_e}\right]. \tag{12d}$$

Substituting Eq. (12a) to Eq. (10), Eq. (10) can be rewritten as

$$\mathbf{U}^{*,i}\left(\mathbf{X}^i\right)=\boldsymbol{\Phi}^{\mathrm{T}}\left(\mathbf{X}^i\right)\boldsymbol{u}^*=\sum_{j=1}^{N_e}\phi_j^i\left(\mathbf{X}^i\right)u^{*,j}, \tag{13}$$

where $\boldsymbol{\Phi}^{\mathrm{T}}\left(\mathbf{X}^i\right)=\boldsymbol{p}^{\mathrm{T}}\left(\mathbf{X}^i\right)\boldsymbol{A}^{-1}\left(\mathbf{X}^i\right)\boldsymbol{B}\left(\mathbf{X}^i\right)$ represents the transfer operator, which includes the shape function values for the i-th Lagrangian point. In the present work, the weight function employs the cubic splines, which can be expressed as

$$W\left(\mathbf{X}^i-\mathbf{x}^j\right)=\begin{cases} e^{-\left(\frac{r_j}{\epsilon}\right)^2} & r_j\le 1,\\ 0, & r_j>1,\end{cases} \tag{14}$$

where $r_j=\left|\mathbf{X}^i-\mathbf{x}^j\right|/H_k$ with $2H_k$ being the length of the support domain in the k-th direction. As discussed in de Tullio and Pascazio [21], the shape functions not only accurately reproduce the linear polynomial but also preserve the partition of unity property $\sum_{j=1}^{N_e}\phi_j^i=1$. In this work, following the previous study [21], the parameters $H_k=1.5h_k$ and $\epsilon=0.3$ are adopted, where h_k denotes the grid spacing of the local fluid mesh in the k-th direction.

Once the velocity is reconstructed at each Lagrangian point, the volume force can be evaluated through

$$\mathbf{F}_b^i=\rho^{n+1}\frac{\boldsymbol{U}_b^i-\mathbf{U}^{*,i}}{\Delta t}, \tag{15}$$

where $\boldsymbol{U}_b^i$ denotes the physical boundary velocity at the i-th Lagrangian point.

To guarantee the conservation of force and torque in the spreading process, a scaling factor is introduced and combined with the same shape functions used in the interpolation process. Consequently, the transfer operator between the Lagrangian and Eulerian restoring forces can be expressed as

$$\mathbf{f}^j = \sum_{i=1}^{N_l} c^i \phi_j^i \mathbf{F}_b^i, \tag{16}$$

where $\mathbf{f}^j$ represents the restoring force at the j-th Eulerian point. N_l is the number of Lagrangian points. c^i denotes the scaling factor related to the i-th Lagrangian point and preserves the following properties as

$$\sum_{j=1}^{N_{e,\text{tot}}} \mathbf{f}^j \Delta V^j = \sum_{i=1}^{N_l} \mathbf{F}_b^i \Delta V^i, \quad \sum_{j=1}^{N_{e,\text{tot}}} \mathbf{x}^j \times \mathbf{f}^j \Delta V^j = \sum_{i=1}^{N_l} \mathbf{X}^i \times \mathbf{F}_b^i \Delta V^i, \tag{17}$$

where $\Delta V^j = \left(h_x^j \times h_y^j \times h_z^j\right)$ denotes the volume of the j-th Eulerian point, $N_{e,\text{tot}}$ is the total number of Eulerian points, and $\Delta V^i = A^i h^i$ represents the volume associated with the i-th Lagrangian point, in which $h^i = 1/3 \sum_{j=1}^{N_e} \phi_i^j \left(h_x^j + h_y^j + h_z^j\right)$. Therefore, the scaling factor can be determined as

$$c^i = \frac{\Delta V^i}{\sum_{j=1}^{N_e} \phi_i^j \Delta V^j}. \tag{18}$$

The intermediate velocity field can be updated by imposing the Eulerian restoring forces on the immersed object, which can be written as

$$\mathbf{u}^{n+1} = \mathbf{u}^* + \Delta t \mathbf{f} / \rho^{n+1}. \tag{19}$$

3.2. The explicit variant MLS-IBM

To reduce the boundary velocity errors in the original MLS-IBM, Chen et al. [14] propose an explicit variant of the original MLS-IBM, where a correction coefficient is introduced into the forcing spreading process. Consequently, the new spreading scheme can be written as:

$$\mathbf{f}^j = Z \sum_{i=1}^{N_l} c^i \phi_j^i \mathbf{F}_b^i, \tag{20}$$

where Z represents the correction coefficient. The correction coefficient can be obtained through minimizing the total error of the Lagrangian restoring force, and it can be defined as

$$Z = \frac{\sum_{k=1}^{N_l} \left[\sum_{j=1}^{N_e} \sum_{i=1}^{N_l} c^i \phi_j^k \phi_j^i F_b^i F_b^k\right]}{\sum_{k=1}^{N_l} \left[\sum_{j=1}^{N_e} \sum_{i=1}^{N_l} c^i \phi_j^k \phi_j^i F_b^i\right]^2}. \tag{21}$$

Although Chen et al. [14] claim that the correction coefficient can reduce the velocity errors, this new spreading scheme disrupts the conservation of force and torque in Eq. (17).

4. The novel implicit moving-least-squares immersed boundary method

Based on the interpolation (Eq. (13)) and spreading (Eq. (16)) schemes employed in the original MLS-IBM, the inequality between the interpolation and spreading procedures can be clearly observed, which indicates that the original MLS-IBM introduces numerical errors in the no-slip boundary condition. The motivation of the present work is to simultaneously eliminate the errors induced by the inequality between the interpolation and spreading operations while maintaining the conservation of force and torque. Moreover, comparisons of errors in the velocity boundary condition, errors in the conservation of force and torque, and computational efficiency between the original MLS-IBM, the explicit variant MLS-IBM, the conventional implicit IBM and the proposed implicit MLS-IBM are conducted.

4.1. The methodology of the implicit MLS-IBM

In this section, the methodology of the proposed implicit MLS-IBM is presented. The spreading process of Eq. (16) can be rewritten as

$$\rho \frac{\mathbf{u}^{n+1,j} - \mathbf{u}^{*,j}}{\Delta t} = \sum_{i=1}^{N_l} c^i \phi_j^i \mathbf{F}_b^i. \tag{22}$$

Substituting Eq. (13) into Eq. (22), Eq. (22) can be rewritten as

$$\frac{\rho}{\Delta t}\left(\boldsymbol{U}_b^k - \sum_{j=1}^{N_e} \phi_j^k \mathbf{u}^{*,j}\right) = \sum_{j=1}^{N_e} \phi_j^k \left(\sum_{i=1}^{N_l} c^i \phi_j^i \mathbf{F}_b^i\right). \tag{23}$$

Eq. (23) can be rewritten in a matrix form as

$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{B}, \tag{24a}$$

$$A_{ki} = \sum_{j=1}^{N_e}\sum_{i=1}^{N_l} \phi_j^k c^i \phi_j^i, \tag{24b}$$

$$B_k = \frac{\rho}{\Delta t}\left(\boldsymbol{U}_b^k - \sum_{j=1}^{N_e} \phi_j^k \mathbf{u}^{*,j}\right), \tag{24c}$$

$$X_i = \mathbf{F}_b^i. \tag{24d}$$

In Eq. (24), assembling the correlation matrix $\mathbf{A}$ consumes substantial computational time for FSI problems with a large number of Lagrangian interpolation points, which is a common issue in both the implicit velocity correction IBM [15,27] and the implicit direct forcing IBM [11]. To improve the computational efficiency of the implicit MLS-IBM, the construction of the correlation matrix $\mathbf{A}$ needs to be optimized. According to Eq. (24b), the coefficient $A_{ki} \neq 0$ requires:

$$\phi_j^k \neq 0 \Rightarrow W\left(\mathbf{X}^k - \mathbf{x}^j\right) \neq 0, \tag{25a}$$

$$\phi_j^i \neq 0 \Rightarrow W\left(\mathbf{X}^i - \mathbf{x}^j\right) \neq 0. \tag{25b}$$

Assuming the fluid point $j \in S(K)$, the relationship $W\left(\mathbf{X}^k - \mathbf{x}^j\right) \neq 0$ can be satisfied. Meanwhile, when the fluid point $j \in S(I)$, $W\left(\mathbf{X}^i - \mathbf{x}^j\right) \neq 0$ can be satisfied. Note that $S(K)$ and $S(I)$ denote two sets of Eulerian points related to the k-th and i-th Lagrangian points, respectively. $A_{ki} \neq 0$ requires these two Lagrangian points k and i to share the common fluid points $j \in S(K) \cap S(I)$. According to Eq. (14), when $j \in S(K)$, $\left\|\mathbf{X}^k - \mathbf{x}^j\right\| \leq \left\|(H_x, H_y, H_z)^{\mathrm{T}}\right\|$; when $j \in S(I)$, $\left\|\mathbf{X}^i - \mathbf{x}^j\right\| \leq \left\|(H_x, H_y, H_z)^{\mathrm{T}}\right\|$.

Consequently, $A_{ki} \neq 0$ requires that the distance between Lagrangian points should satisfy the following relationship:

$$\begin{aligned}\left\|\mathbf{X}^k - \mathbf{X}^i\right\| &= \left\|\left(\mathbf{X}^k - \mathbf{x}^j\right) - \left(\mathbf{X}^i - \mathbf{x}^j\right)\right\| \\ &\leq \left\|\mathbf{X}^k - \mathbf{x}^j\right\| + \left\|\mathbf{X}^i - \mathbf{x}^j\right\| \\ &\leq \left\|(2H_x, 2H_y, 2H_z)^{\mathrm{T}}\right\| = \sqrt{4H_x^2 + 4H_y^2 + 4H_z^2}.\end{aligned} \tag{26}$$

Assuming there are N_i Lagrangian points surrounding the k-th Lagrangian point within a distance of $\sqrt{4H_x^2 + 4H_y^2 + 4H_z^2}$, Eqs. (21) and (24b) can be rewritten as

$$Z = \frac{\sum_{k=1}^{N_l}\left[\sum_{j=1}^{N_e}\sum_{i=1}^{N_i} c^i \phi_j^i \phi_j^k F_b^i F_b^k\right]}{\sum_{k=1}^{N_l}\left[\sum_{j=1}^{N_e}\sum_{i=1}^{N_i} c^i \phi_j^k \phi_j^i F_b^i\right]^2}, \tag{27}$$

$$A_{ki} = \sum_{j=1}^{N_e}\sum_{i=1}^{N_i} \phi_j^k c^i \phi_j^i, \tag{28}$$

respectively. Eq. (28) significantly improves the computational efficiency of the implicit MLS-IBM.

The above linear system (Eq. (24)) can be efficiently solved by the generalized minimum residual (GMRES) algorithm with the compress row technique to reduce memory usage. Note that the incomplete LU decomposition is adopted as the preconditioner for the GMRES method. Once the restoring forces at Lagrangian points are obtained, the restoring force at surrounding Eulerian points can be determined by the spreading scheme of Eq. (16), and the intermediate velocity can then be updated through Eq. (19). The errors induced by the inequality between the interpolation and spreading procedures in the original MLS-IBM can be eliminated by the present implicit scheme. Furthermore, the implicit MLS-IBM preserves the conservation of force and torque. Note that the proposed implicit framework and the optimization technique for correlation matrix assembling are also suitable for the regularized delta function in conventional IBMs [7,28].

4.2. Examinations of boundary condition enforcement and conservation capability

In this subsection, the numerical test of flow past a cylinder is conducted. The diameter of the cylinder is D and the Reynolds number is set to $Re = \rho U_0 D/\mu = 40$. The computational domain size is set to $[-1.5D, 1.5D] \times [1.5D, 1.5D]$ and is uniformly discretized with a mesh spacing of $h = 0.05D$. The grid spacing between Lagrangian points is also set to h. To evaluate whether the no-slip boundary condition is accurately enforced, the velocity error in the x direction at each Lagrangian point is defined as $\varepsilon_u = |U_b - \mathrm{U}_{b,\mathrm{MLS}}|/U_0$, where $\mathrm{U}_{b,\mathrm{MLS}}$ denotes the boundary velocity reconstructed by the MLS method in the x direction. To assess whether the conservation of force and torque is maintained within different MLS-IBMs, the errors in the conservation of force and torque can be defined according to Eq. (17) as

$$\varepsilon_{\mathrm{F}_x} = \left|\sum_j \mathrm{f}_x\left(\mathbf{x}^j\right)\Delta V_j - \sum_i \mathrm{F}_{b,x}\left(\mathbf{X}^i\right)\Delta V_i\right|, \tag{29a}$$

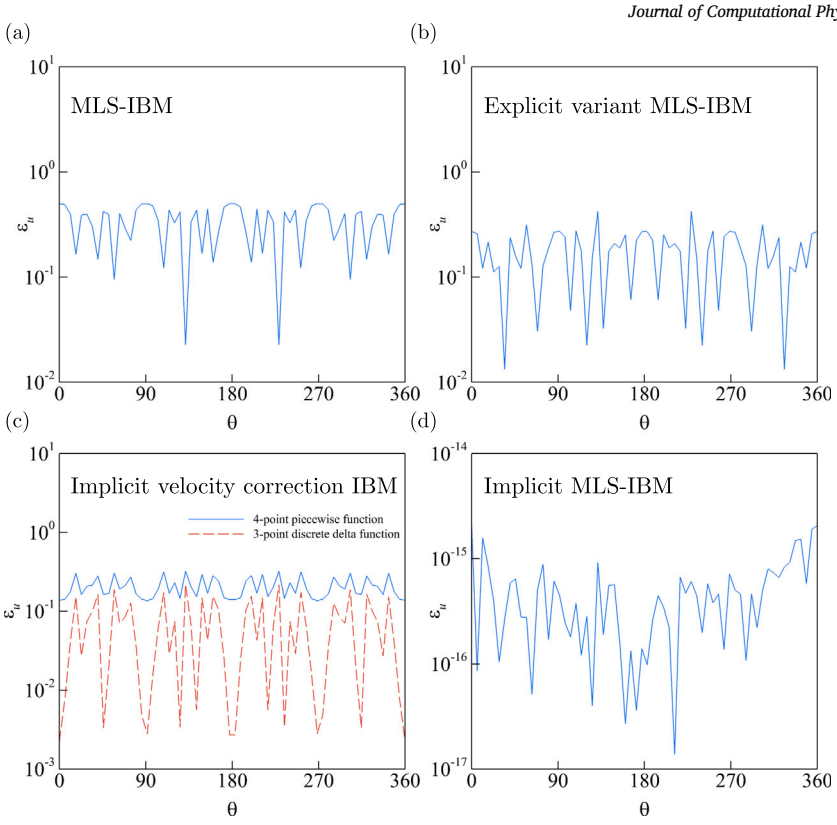

Fig. 3. Boundary velocity errors in the x direction for (a) the MLS-IBM [20], (b) the explicit variant MLS-IBM [14], (c) the implicit velocity correction IBM [15] and (d) the implicit MLS-IBM.

$$\varepsilon_{\mathrm{F}_y} = |\sum_j \mathrm{f}_y\left(\mathbf{x}^j\right)\Delta V_j - \sum_i \mathrm{F}_{b,y}\left(\mathbf{X}^i\right)\Delta V_i|, \tag{29b}$$

$$\varepsilon_t = |\sum_j \mathbf{x}^j \times \mathbf{f}\left(\mathbf{x}^j\right)\Delta V_j - \sum_i \mathbf{X}_i \times \mathbf{F}_b\left(\mathbf{X}^i\right)\Delta V_i|. \tag{29c}$$

The initial condition of the whole domain is set as $(\rho, u, v) = (1.0, 0.1, 0.0)$. In the first time step, the difference between the desired and reconstructed boundary velocities is largest. Therefore, the boundary velocity errors are collected after the IBM is applied at this initial step. To facilitate comparison with other implicit IBMs, the results generated by the implicit velocity correction IBM are also included.

Fig. 3 shows the distributions of boundary velocity errors on the cylinder surface. Compared with the MLS-IBM and the explicit variant MLS-IBM, the implicit MLS-IBM can effectively suppress the boundary velocity errors, which are close to machine accuracy on such coarse meshes, indicating that the no-slip boundary condition is accurately enforced on the immersed boundary. Note that the explicit variant MLS-IBM only slightly reduces the boundary velocity errors. Although the implicit velocity correction IBM can accurately enforce the velocity boundary condition within the framework of the regularized delta function for interpolation and spreading operations, the boundary velocity reconstructed from the corrected flow field using the MLS method shows a significant discrepancy from the desired boundary condition. Therefore, in contrast to previous implicit IBMs based on the regularized delta function, the present implicit MLS-IBM ensures that the reconstructed boundary value has specific accuracy order. Note that, for the LBM framework, there is no velocity projection step to satisfy the divergence free condition [17,29]. Therefore, the no-slip boundary condition should be accurately enforced by the IBM, with the boundary velocity errors close to machine accuracy; otherwise, the boundary velocity errors will induce mass penetration in LBM framework.

Table 1 shows the errors in the conservation of force and torque after implementing different MLS-IBMs for 100 time steps. It is evident from Table 1 that the proposed implicit MLS-IBM can preserve the conservation of force and torque. Meanwhile, the error of $\varepsilon_{\mathrm{F}_x}$ in the explicit variant MLS-IBM is two orders of magnitude greater than that of the other two methods, because the correction coefficient in the spreading operation breaks the conservation of force and torque. In summary, the proposed implicit MLS-IBM not only eliminates the boundary velocity errors but also maintains the conservation of force and torque.

Table 1
The errors of force conservation in the x direction, force conservation in the y direction, and torque conservation.

Error	ε_{F_x}	ε_{F_y}	ε_t
MLS-IBM	3.661×10^{-7}	2.640×10^{-13}	6.404×10^{-13}
Explicit variant MLS-IBM	3.284×10^{-5}	3.156×10^{-12}	4.560×10^{-13}
Implicit MLS-IBM	3.724×10^{-7}	2.577×10^{-13}	6.374×10^{-13}

4.3. Comparisons of time consumption

To clearly compare the computational efficiency of the MLS-IBM, the explicit variant MLS-IBM and the implicit MLS-IBM, the computational sequences for these three approaches are presented in Algorithms 1, 2 and 3, respectively. In this subsection, the flow past a stationary and rotating sphere is conducted to test the computation efficiency for stationary and moving problems, respectively. The computational domain is set to $[-10,10]\times[-10,10]\times[-10,10]$, where the flow region around the sphere $[-0.5,0.5]\times[-0.5,0.5]\times[-0.5,0.5]$ is discretized by a uniform mesh with mesh spacing of $h=1/200$. The sphere is uniformly discretized by Lagrangian points. Based on the conventional implicit IBMs [29,30], the averaged Lagrangian spacing should be around h and the radius of the sphere is dynamically adjusted. In all numerical tests, the convergence criteria for the proposed implicit MLS-IBM with the GMRES method is set to 10^{-17} to maintain the boundary velocity errors within machine round-off. The reported computational time is averaged by implementing different IBMs 100 times to minimize random machine errors.

Algorithm 1 Original MLS-IBM.

A1: Calculate the transfer operator $\mathbf{\Phi}$;
A2: Reconstruct the flow velocity $\mathbf{U}^*$ at Lagrangian points;
A3: Calculate the restoring forces at Lagrangian points by Eq. (15);
A4: Spread the Lagrangian restoring forces to Eulerian points by Eq. (16);
A5: Update the flow velocity through Eq. (19).

Algorithm 2 Explicit variant of MLS-IBM

A1: Calculate the transfer operator $\mathbf{\Phi}$;
A2: Reconstruct the flow velocity $\mathbf{U}^*$ at Lagrangian points;
A3: Calculate the restoring forces at Lagrangian points by Eq. (15);
B1: Calculate the correction coefficient Z by Eq. (21);
B2: Spread the Lagrangian restoring forces to Eulerian points by Eq. (20);
A5: Update the flow velocity through Eq. (19).

Algorithm 3 Implicit MLS-IBM

A1: Calculate the transfer operator $\mathbf{\Phi}$;
C1: Calculate the correlation matrix $\mathbf{A}$ and $\mathbf{A}^{-1}$ (for stationary problems);
A2: Reconstruct the flow velocity $\mathbf{U}^*$ at Lagrangian points;
C2: Calculate the restoring forces at Lagrangian points by Eq. (24a);
A4: Spread the Lagrangian restoring forces to Eulerian points by Eq. (16);
A5: Update the flow velocity through Eq. (19).

For stationary problems, the transfer operator $\mathbf{\Phi}$ and the correlation matrix $\mathbf{A}$ only need to be generated once. Therefore, the computational time consumed by the original MLS-IBM, the explicit variant MLS-IBM and the implicit MLS-IBM are A2 + A3 + A4 + A5, A2 + A3 + B1 + B2 + A5 and A2 + C2 + A4 + A5, respectively. For stationary problems, the C2 step in Algorithm 3 can be implemented through two schemes, namely, matrix manipulation of $\mathbf{X}_{(N_l\times1)}=\mathbf{A}^{-1}_{(N_l\times N_l)}\mathbf{B}_{(N_l\times1)}$ (Scheme I) and using GMRES to solve $\mathbf{A}_{(N_l\times N_l)}\mathbf{X}_{(N_l\times1)}=\mathbf{B}_{(N_l\times1)}$ (Scheme II). In Table 2, the computational time for the MLS-IBM, the explicit variant MLS-IBM and the implicit MLS-IBM for stationary problems are presented. It can be seen that these two schemes of the proposed implicit MLS-IBM are more efficient than the explicit variant MLS-IBM. Although the original MLS-IBM is the fastest approach among these methods, the MLS-IBM cannot accurately enforce the no-slip boundary condition, as discussed in Section 4.2. These results demonstrate that the proposed MLS-IBM is more suitable for simulating stationary FSI problems with complex geometries.

To further evaluate the computation complexities of these IBMs, the scaling relationships between the CPU time and the number of Lagrangian points for different IBMs are presented in Fig. 4. It is evident from Fig. 4 that the computational complexities of the MLS-IBM, the explicit variant MLS-IBM, the implicit MLS-IBM (scheme I) and the implicit MLS-IBM (scheme II) are $O(N_l^{1.388})$, $O(N_l^{1.070})$, $O(N_l^{1.917})$ and $O(N_l^{1.185})$, respectively. Based on the computational complexities, directly solving the implicit system (scheme I) is not suitable for FSI problems with a large number of Lagrangian points (e.g., $N_l>2\times10^4$). Meanwhile, the computational complexity of the implicit MLS-IBM with GMRES method (scheme II) is similar to that of the previous MLS-IBMs, indicating that the implicit MLS-IBM with GMRES method is also suitable for stationary problems with a large number of Lagrangian points. Nonetheless, when

Table 2
The computational time consumed by the IBM part for stationary problems in one time step, which is performed on one CPU core (AMD Ryzen™ 9 5950X, 3.4 GHz).

Number of Lagrangian points	5000	10000	15000	20000	25000	30000	40000
MLS-IBM	0.001 s	0.003 s	0.006 s	0.007 s	0.011 s	0.013 s	0.018 s
Explicit variant MLS-IBM	0.208 s	0.461 s	0.680 s	0.956 s	1.267 s	1.453 s	1.959 s
Implicit MLS-IBM (Scheme I)	0.020 s	0.070 s	0.150 s	0.268 s	0.410 s	0.599 s	1.074 s
Implicit MLS-IBM (Scheme II)	0.055 s	0.108 s	0.178 s	0.253 s	0.330 s	0.423 s	0.654 s

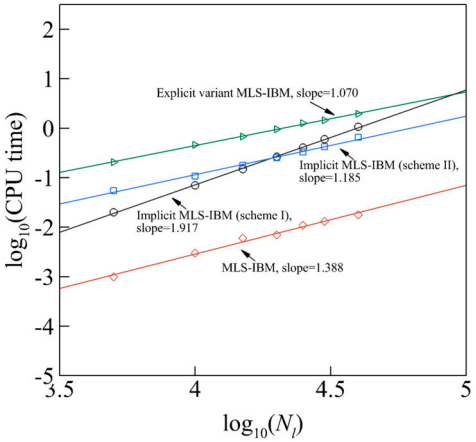

Fig. 4. The average CPU time consumed by different IBMs for one time step versus the number of Lagrangian points in simulating stationary problems.

Table 3
The computational time consumed by the IBM part for moving problems in one time step, which is performed on one CPU core (AMD Ryzen™ 9 5950X, 3.4 GHz).

Number of Lagrangian points	5000	10000	15000	20000	25000	30000	40000
MLS-IBM	0.009 s	0.019 s	0.031 s	0.037 s	0.052 s	0.058 s	0.095 s
Explicit variant MLS-IBM	0.214 s	0.514 s	0.782 s	1.093 s	1.336 s	1.593 s	2.005 s
Implicit MLS-IBM	0.278 s	0.581 s	0.915 s	1.276 s	1.726 s	2.032 s	2.942 s

solving such problems, the whole computational procedure should be parallelized using various parallel techniques, such as OpenMP, MPI and GPU parallel techniques, to accelerate computational speed. It is worth noting that the above scaling discussion is correlated to the real efficiency comparison when the number of Lagrangian points N_l is sufficiently large. For small N_l, as shown in Fig. 4, the absolute efficiency of different methods might be less relevant to the scaling.

The multi-direct forcing IBM is another popular approach to achieve low boundary velocity errors. However, Wang et al. [31] demonstrate that, to achieve a similar velocity convergence criteria, the computational efficiency of the conventional implicit IBM with the GMRES method is more efficient than that of the multi-direct forcing IBM. For instance, Wang et al. [31] confirm that, for the multi-direct forcing IBM with $\varsigma_r < 10^{-6}$ and the implicit IBM with $\varsigma_r < 10^{-10}$, the implicit IBM with the GMRES method is about 26 times faster than the multi-direct forcing scheme with 10^3 Lagrangian points, where $\varsigma_r = \frac{1}{U_{ref}}\sqrt{\frac{1}{N_l}\sum_{i=1}^{N_L}\left|\boldsymbol{U}_b^i - \mathbf{U}^{*,i}\right|^2}$ denotes the velocity error convergence criteria. Considering that the correlation matrix assembly is optimized in the proposed implicit MLS-IBM, the proposed implicit MLS-IBM is more efficient than the conventional implicit IBMs. Moreover, Zhao et al. [30] demonstrate that the computational complexity of the multi-direct forcing IBM is $O(N_l^{1.683})$, which is much higher than that of the proposed implicit MLS-IBM ($O(N_l^{1.185})$). Given that the MLS method is more complex than the regularized delta function, the original MLS-IBM with multi-direct forcing scheme is less efficient than the multi-direct forcing IBM. Therefore, it can be concluded that to achieve a similar velocity error convergence criteria, the proposed implicit MLS-IBM is more efficient than the original MLS-IBM with the multi-direct forcing scheme.

For moving problems, the position of the immersed object is updated at each time step, necessitating the recalculation of the transfer operator $\mathbf{\Phi}$ and the correlation matrix $\mathbf{A}$ accordingly. Therefore, these three approaches have to go through whole processes presented in Algorithms 1, 2 and 3, respectively. It is evident from Table 3 that the computational efficiency of the proposed implicit MLS-IBM is comparable to that of the explicit variant MLS-IBM. Although the original MLS-IBM is the most efficient among these three methods, it contains the errors induced by the inequality between the interpolation and spreading processes. To evaluate the computational complexities of these methods, the scaling relationships between the CPU time and the number of Lagrangian points for different IBMs are presented in Fig. 5. It can be seen that the computational complexity of the proposed implicit MLS-IBM is

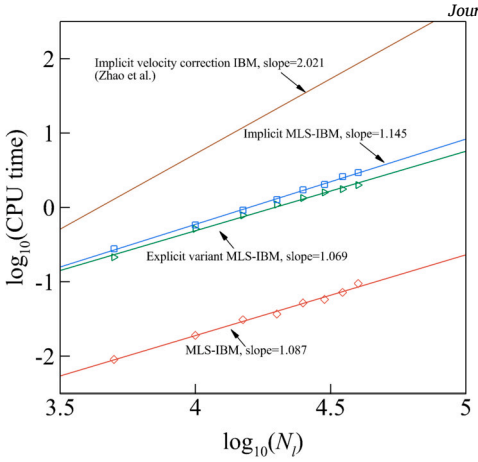

Fig. 5. The average CPU time consumed by different IBMs for one time step versus the number of Lagrangian points in simulating moving problems, where the regression line for the implicit velocity correction IBM is provided by Zhao et al. [30].

Table 4
The advantages and disadvantages of different MLS-IBMs.

Method	Advantages	Disadvantages
MLS-IBM	Explicit Simplest Fastest Preserve the conservation	Largest boundary velocity error
Explicit variant MLS-IBM	Explicit Partially alleviate boundary velocity errors	Break the conservation Inefficient for stationary problems
Implicit MLS-IBM	Implicit Highest accuracy Lowest boundary velocity errors Preserve the conservation More efficient than conventional implicit IBMs	Slight slow for moving problems

$O(N_l^{1.145})$, which is comparable to the computational complexities of the original MLS-IBM $O(N_l^{1.087})$ and the explicit variant MLS-IBM $O(N_l^{1.069})$. In addition, Zhao et al. [30] demonstrate that, using the regularized delta function, the computational complexity of the implicit velocity correction IBM is $O(N_l^{2.021})$. Therefore, the proposed implicit MLS-IBM is much more efficient than the conventional implicit IBMs, because Eq. (28) significantly optimizes the assembly of the correlation matrix. Considering the accuracy of boundary condition enforcement, conservation of force and torque, and computational complexity, the proposed implicit MLS-IBM can be a better solution for practical FSI problems with moving boundaries. Although the computational complexity of the proposed implicit MLS-IBM is significantly lower than conventional implicit IBMs, simulating complex moving problems with a large number of Lagrangian points still requires substantial computational time. In such cases, the whole computational procedure can be accelerated by using various parallel techniques, such as OpenMP, MPI and GPU parallel techniques. Specifically, the proposed implicit system can be efficiently solved using the parallel GMRES solver on GPU platforms [32].

Lastly, the advantages and disadvantages of different MLS-IBMs are summarized in Table 4.

5. Results and discussion

In this section, the space and time accuracy of the proposed implicit MLS-IBM, integrated with the flow solver (LBFS), is first examined through the flow around a circular cylinder immersed in a lid-driven cavity. Subsequently, the implicit MLS-IBM is employed to simulate the self-propulsion of a 2D unconstrained flapping foil, the flow past a 3D torus, and the flow past a flexible flag, demonstrating the capability and accuracy of the proposed implicit MLS-IBM in tackling complex FSI problems with complex geometries, moving boundaries and large deformations.

5.1. Space and time accuracy test

In this subsection, the global spatial and temporal accuracy of the proposed implicit MLS-IBM with the LBFS are examined by simulating the flow around a circular cylinder immersed in a lid-driven cavity (see Fig. 6(a)), where the square computation domain is set to $[0, L] \times [0, L]$ and the diameter of the cylinder is $0.4L$. The Reynolds number of $Re = U_{lid} L/\nu = 1000$ is applied, where U_{lid} denotes the velocity of the top boundary. The square computational domain is discretized with six different uniform meshes with

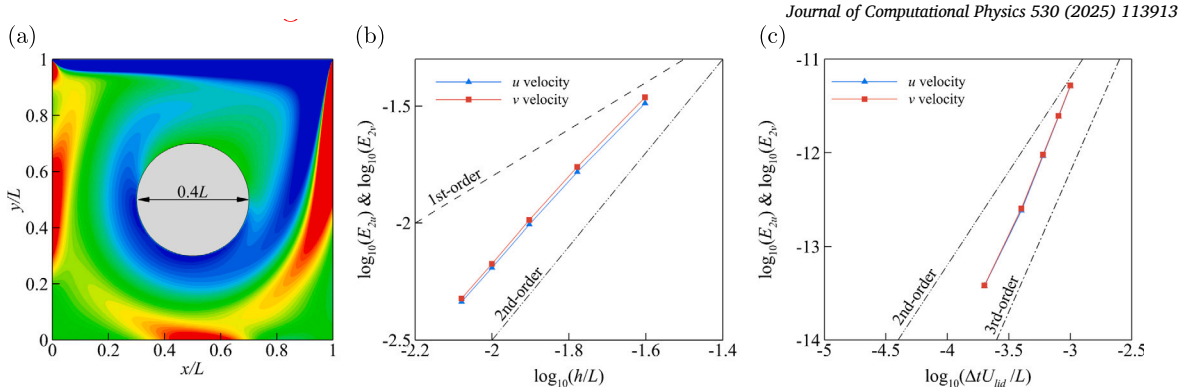

Fig. 6. (a) Schematic view of the flow around a circular cylinder immersed in a lid-driven cavity; convergence study in (b) space and (c) time of the proposed implicit MLS-IBM.

grid size of $h = L/40, L/60, L/80, L/100, L/120, L/360$ to investigate the convergence order. The no-slip boundary condition on the cylinder surface is enforced by the proposed implicit MLS-IBM. Given that the exact steady solution of this numerical test is unknown, the numerical results obtained with the finest mesh ($h = L/360$) are adopted as the reference solution. The relative errors E_{2u} and E_{2v} are obtained at $t^* = U_{lid}t/L = 200$, and the errors are defined as

$$E_{2u} = \sqrt{\frac{\sum_{i=1}^{N_E}\left[\frac{u_i^{\text{num}} - u_i^{\text{ref}}}{U_{lid}}\right]^2}{N_E}}, \quad E_{2v} = \sqrt{\frac{\sum_{i=1}^{N_E}\left[\frac{v_i^{\text{num}} - v_i^{\text{ref}}}{U_{lid}}\right]^2}{N_E}}, \tag{30}$$

where the superscripts "num" and "ref" denote the numerical solutions and reference results, respectively. Note that the bilinear interpolation method is employed to interpolate the reference solution from the finest mesh for the coarse mesh. Chen et al. [14] demonstrate that the original MLS-IBM with multi-direct forcing scheme and the explicit variant of MLS-IBM integrated with a second-order flow solver can only achieve first-order spatial accuracy. In contrast, as shown in Fig. 6(b) that the global spatial accuracy of the proposed implicit MLS-IBM is around second-order, which is higher than the accuracy order of previous MLS-IBMs presented in Chen et al. [14]. It is noted that the proposed implicit MLS-IBM not only eliminates the boundary velocity errors but also maintains the conservation of force and torque. Based on the time accuracy test of IBM in previous study [33], six time steps $\Delta t = 1\times10^{-2}, 8\times10^{-3}, 6\times10^{-3}, 4\times10^{-3}, 2\times10^{-3}, 6.25\times10^{-4}$ are considered, and the errors are calculated at the fixed mesh spacing $h = L/100$. The numerical result predicted with the finest time step $\Delta t = 6.25\times10^{-4}$ is adopted as the reference solution. It is evident from Fig. 6(c) that the time accuracy of the proposed implicit MLS-IBM integrated with the LBFS is around third-order, which is the desired accuracy order with the adopted SSP-RK method. These numerical accuracy tests demonstrate that the proposed implicit MLS-IBM can preserve sufficient space and time accuracy, which can be further extended to the high-order numerical framework in the future.

To evaluate whether the proposed implicit MLS-IBM can maintain mass conservation, the time evolution of the maximum magnitude of the velocity divergence ($|\nabla\cdot\mathbf{u}|_{\text{b,max}}$) at Lagrangian points is recorded in the space accuracy test, where $|\nabla\cdot\mathbf{u}|_{\text{b}}$ is interpolated from the velocity divergence field after implementing the implicit MLS-IBM. Previous studies have demonstrated that the velocity divergence value is of the order of $O(Ma^2)$ and the incompressibility condition can be recovered at $Ma < 0.3$ [34–36]. It is evident from Fig. 7 that, even on the coarsest mesh ($h = L/40$), the maximum velocity divergence value is around 8.8×10^{-5} during the simulation, indicating that the local maximum Ma is less than 0.01, which is much smaller than the suggested Mach number of 0.3 in previous studies [34–36]. Therefore, the proposed implicit MLS-IBM can well preserve the mass conservation.

5.2. Propulsion of a 2D unconstrained flapping foil

In this subsection, the proposed implicit MLS-IBM is employed to simulate the propulsive performance of a 2D flapping foil, where the 2D flapping foil propels throughout the computational domain. This FSI problem involves moving boundaries and nonlinear vibrations, making it very challenging for conventional CFD approaches using body-fitted meshes. The geometry of the 2D flapping foil is illustrated in Fig. 8, where c and l denote the chord length and the thickness, respectively.

In this problem, the 2D flapping foil pitches around the center (see the red point in Fig. 8), and the governing equation of the pitching motion can be written as

$$\theta(t) = \theta_m \sin(2\pi f t), \tag{31}$$

where θ_m and f denote the maximum pitching angle and the pitching frequency, respectively. The propulsive behavior is governed by the Newton's second law, which can be written as

$$M\frac{d^2\mathbf{X}}{dt^2} = \mathbf{F}_f, \tag{32}$$

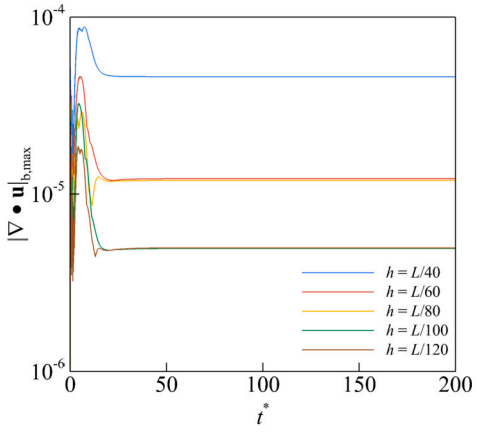

Fig. 7. The time evolution of the maximum magnitude of the velocity divergence ($|\nabla \cdot \mathbf{u}|_{\mathrm{b,max}}$) at Lagrangian points.

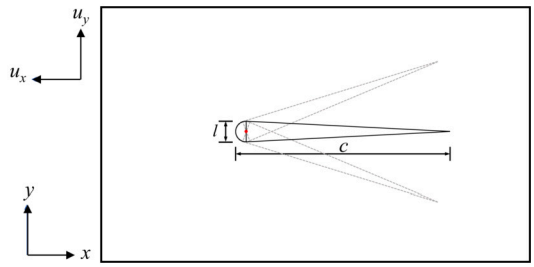

Fig. 8. Schematic view of the 2D unconstrained flapping foil, where the foil consists of an arc and a triangle. c and l denote the chord length and thickness, respectively.

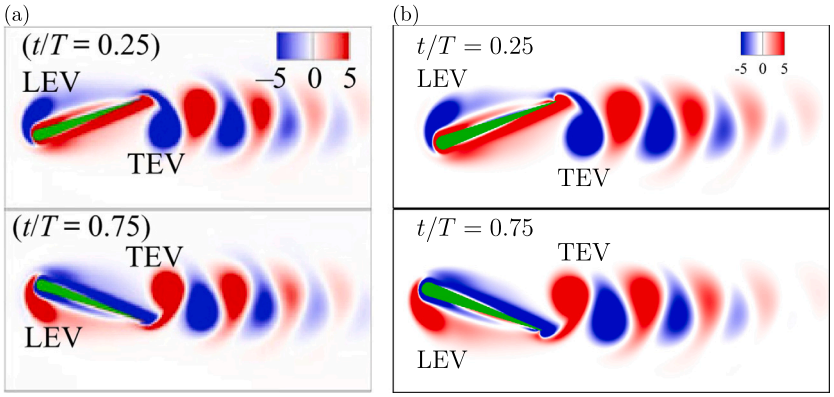

Fig. 9. (a) The instantaneous vorticity contours adopted from the reference [37]; (b) the instantaneous vorticity contours generated by the proposed implicit MLS-IBM. The pitching parameters are $f = 1$ and $\theta_m = 20^o$.

where $M = \rho_s S$ represents the mass of the flapping foil, $\mathbf{F}_f$ denotes the total fluid force exerted on the foil, and $\mathbf{X}$ is the position of the pivot point. The fluid-structure interaction approach is presented in Appendix A. The propulsive performance is controlled by the following dimensionless parameters: mass ratio $m = M/M_f$, where $M_f = \rho S$ is the fluid mass with equivalent area, Reynolds number $Re = \rho U c/\mu$, the thickness-chord ratio l/c, the maximum pitching angle θ_m and the pitching frequency f. In this work, the parameters $m = 1$, $Re = 200$ and $l/c = 0.1$ are adopted. The size of the computational domain is $[-15c, 10c] \times [-5c, 5c]$, which is discretized by a non-uniform mesh. The propulsive region of $[-14c, 2c] \times [-2c, 2c]$ is discretized by a uniform mesh with a grid spacing of $h = 0.01c$. The surface of the 2D flapping foil is discretized by 208 Lagrangian points.

The comparisons of the instantaneous vorticity contours of the 2D flapping foil between the results of Lin et al. [37] and those predicted by the implicit MLS-IBM are presented in Fig. 9. It is evident from Fig. 9 that the present results are in a good agreement with the reference contours of Lin et al. [37], where the leading-edge vortex (LEV) and the trailing-edge vortex (TEV) of the flapping foil are accurately captured, qualitatively indicating the capability of the proposed implicit MLS-IBM in simulating complex FSI problems with moving boundaries.

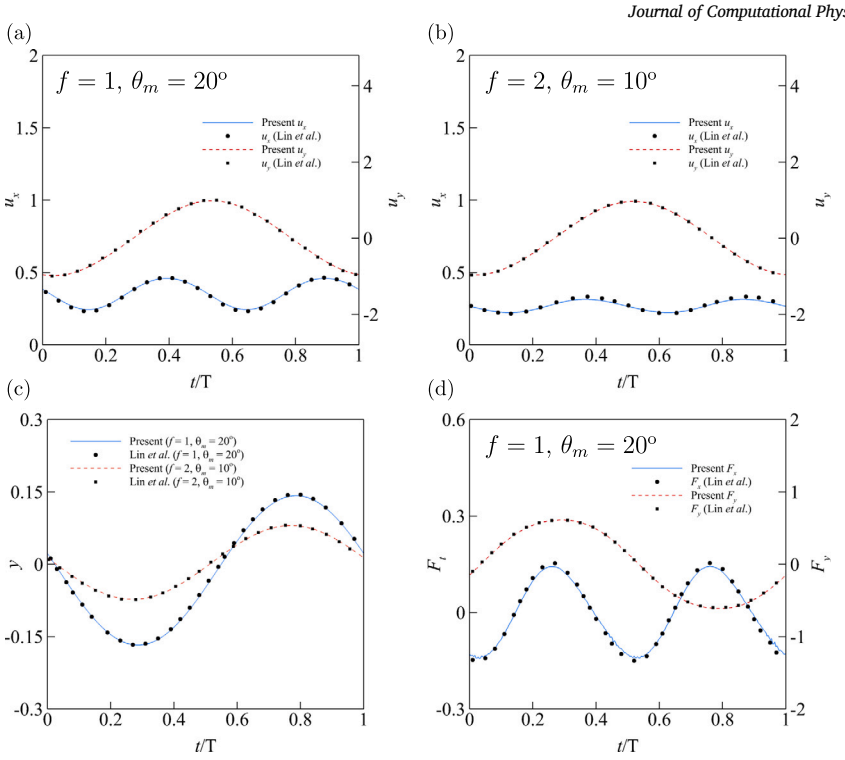

Fig. 10. (a) The time evolutions of the propulsive speeds at $f = 1$ and $\theta_m = 20^o$; (b) the time evolutions of the propulsive speeds at $f = 2$ and $\theta_m = 10^o$; (c) the time evolutions of the pivot point position; (d) the time evolutions of the hydrodynamic forces at $f = 1$ and $\theta_m = 20^o$.

To further quantitatively examine the accuracy and feasibility of the implicit MLS-IBM, comparisons of time histories of the propulsive velocities, lateral position of the pivot point and hydrodynamic forces are shown in Fig. 10. It can be seen that the time evolutions of propulsive speeds u_x and u_y, the lateral position y of the pivot point, and the hydrodynamic forces F_t and F_y agree well with the reference data, demonstrating that the proposed implicit MLS-IBM can accurately capture the nonlinear dynamic characteristics of FSI problems with moving boundaries. These numerical results further illustrate that the implicit MLS-IBM is capable of accurately predicting the flow dynamics around the moving object.

5.3. Flow past a 3D torus

In this subsection, to examine the capability of the proposed implicit MLS-IBM for simulating FSI problems with complex geometries, the flow past a 3D torus is investigated. Fig. 11 shows the geometry of the 3D torus, where d and D represent the cross-section diameter and the mean diameter of the 3D torus, respectively. The aspect ratio of the 3D torus is set as $Ar = D/d = 2$. The flow dynamics are governed by the Reynolds number $Re = \rho U_0 d/\mu$, where U_0 is the free stream velocity. In this numerical test, the flow direction aligns with the x-axis. There are 2016 Lagrangian points for discretizing the surface of the 3D torus. The size of the computational domain is $[-10d, 15d] \times [-10d, 10d] \times [-10d, 10d]$, where a non-uniform mesh of $101 \times 121 \times 121$ is employed to discretize the computational domain. The center of the 3D torus is located at the origin $(0, 0, 0)$ and the fluid region $(1.3d \times 3.6d \times 3.6d)$ around the 3D torus is discretized by a uniform mesh of $27 \times 73 \times 73$. The drag force coefficient is defined as $C_d = F_x/(1/2\rho U_0^2 \pi D d)$.

Figs. 12 and 13 show the streamlines for flow past the 3D torus at $Re = 40$ and 93. It can be seen that, at $Re = 40$, the recirculation region along the axis forms away from the rear part of torus, which is different from the wake phenomena of a bluff body without a hole; at $Re = 93$, the recirculation region behind the cross-section of torus becomes larger, and the distance between the axial recirculation region and the rear of the 3D torus increases. In addition, the size of the axial recirculation region is further reduced. These observations are consistent with those in the reference [27], qualitatively indicating that the proposed implicit MLS-IBM can accurately capture the flow dynamics around complex geometries. To quantitatively assess the accuracy of the implicit MLS-IBM, the comparison of drag coefficient between present results and reference data are presented in Table 5. It is evident from Table 5 that the numerical results predicted by the implicit MLS-IBM agree well with the reference data, demonstrating that the implicit MLS-IBM can accurately provide aerodynamic forces.

5.4. Flow past a 3D flexible flag

In this subsection, to examine the capability and accuracy of the proposed implicit MLS-IBM for simulating complex FSI problems involving moving boundaries and large deformations, the flow past a 3D flexible flag is simulated [21,40–42]. Fig. 14 shows the

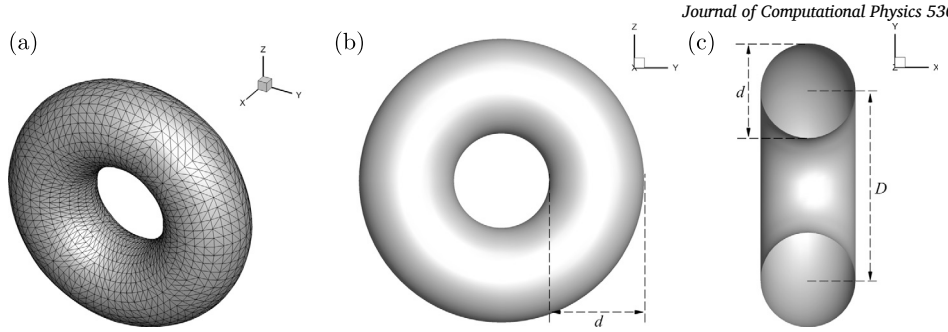

Fig. 11. (a) The geometry of the 3D torus; (b) front view and (c) side view.

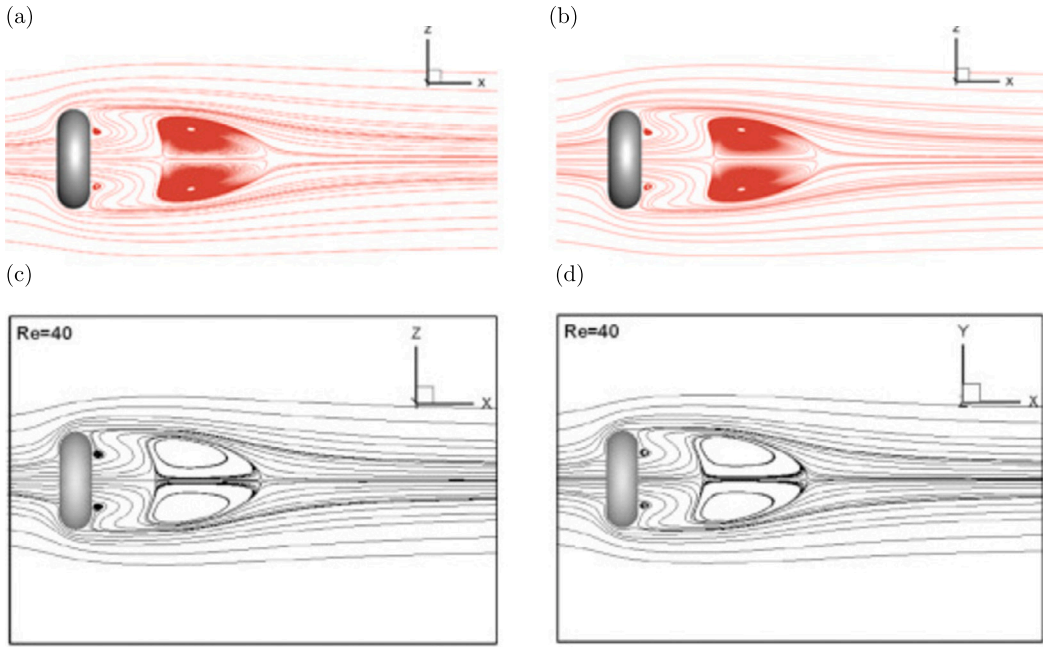

Fig. 12. The streamlines for flow past a 3D torus at $Re = 40$; upper row: present results, lower row: reference results [27].

Table 5
Comparison of drag coefficient for the flow past 3D stationary torus at $Re = 40$ and 93.

	$Re = 40$	$Re = 93$
Wang et al. [38]	1.318	0.988
Yang et al. [39]	1.324	0.983
Wu and Shu [27]	1.335	1.013
Implicit MLS-IBM	1.343	1.047

geometry of the 3D flexible square flag, where the length of the flag is L and the flow direction aligns with the x axis. As shown in Fig. 14, 2025 Lagrangian points are employed to evenly discretize the surface of the 3D flag. The leading edge of the 3D flag is pinned, and the other three edges can freely move. The nonlinear dynamic behavior of the 3D flexible is governed by the following equation [41] as

$$\rho_s \frac{\partial^2 \mathbf{X}}{\partial t^2} = \sum_{i,j=1}^{2} \left[\frac{\partial}{\partial s_i} \left(\sigma_{ij} \frac{\partial \mathbf{X}}{\partial s_j} \right) - \frac{\partial^2}{\partial s_i \partial s_j} \left(\kappa_{ij}^b \frac{\partial^2 \mathbf{X}}{\partial s_i \partial s_j} \right) \right] + \mathbf{F}_f, \tag{33}$$

where s_1 and s_2 denote the local curvilinear coordinate. ρ_s, $\mathbf{X}$, $\mathbf{F}_f$ are the area density of the 3D flexible flag, the position of Lagrangian points on the flag surface, and the fluid force exerted by surrounding flows, respectively. σ_{ij} is defined as

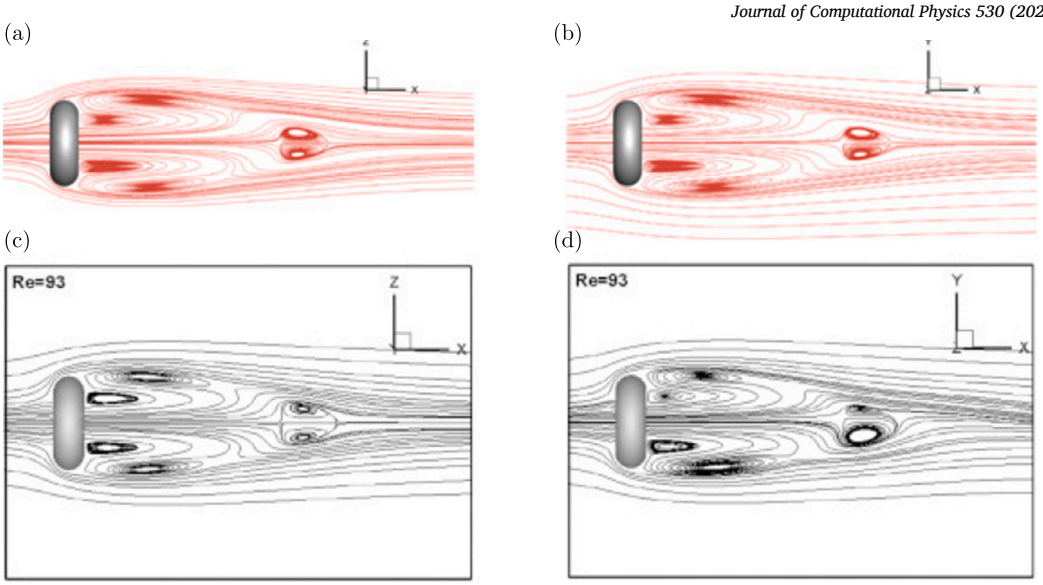

Fig. 13. The streamlines for flow past a 3D torus at $Re = 93$; upper row: present results, lower row: reference results [27].

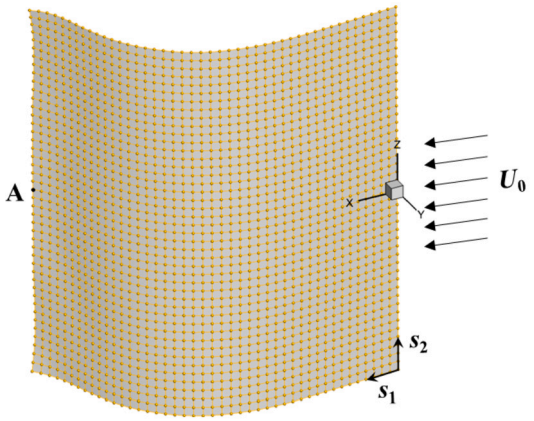

Fig. 14. Schematic view of the 3D flapping flexible plate.

$$\sigma_{ij} = \varphi_{ij}\left(\frac{\partial \mathbf{X}}{\partial s_i}\cdot\frac{\partial \mathbf{X}}{\partial s_j} - \Gamma^0_{ij}\right), \tag{34a}$$

$$\Gamma^0_{ij} = \begin{cases} 1, \text{if } i = j, \\ 0, \text{if } i \neq j. \end{cases} \tag{34b}$$

If $i = j$, φ_{ij} and κ^b_{ij} represent the tension and bending coefficients, respectively; if $i \neq j$, φ_{ij} and κ^b_{ij} denote the shearing and twisting coefficients, respectively. In the present work, the numerical approach for solving Eq. (33) is adopted from the work of Ma et al. [43]. The fluid-structure interaction approach is shown in Appendix A.

This problem is governed by the following dimensionless parameters: the Reynolds number $Re = \rho U_0 L/\mu$, the mass ratio $\Theta = \rho_s/(\rho L)$, the dimensionless stretching or shearing coefficients $\Phi_{ij} = \varphi_{ij}/(\rho_s U_0^2)$, and the dimensionless bending or twisting coefficients $K_{ij} = \kappa^b_{ij}/(\rho_s U_0^2 L^2)$. In this numerical test, the dimensionless parameters are set to $Re = 200$, $\Theta = 1$, $\Phi_{11} = \Phi_{22} = 10^3$, $\Phi_{12} = 10$, and $K_{11} = K_{22} = K_{12} = 10^{-4}$ for easy comparison with the reference. The size of the computational domain is $[-2L, 7L] \times [-4L, 4L] \times [-L, L]$, and the computational domain is discretized by a non-uniform mesh of $315 \times 150 \times 120$. The center of the leading edge is fixed at the origin of the domain, and the fluid domain $[-0.2L, 1.3L] \times [-0.7L, 0.7L] \times [-0.7L, 0.7L]$ around the 3D flexible flag is discretized by a uniform mesh of $113 \times 76 \times 91$ to accurately simulate the complex interactions between surrounding flows and the flexible flag. The boundary conditions for the computational domain are: the Dirichlet boundary conditions ($u = U_0$, $v = w = 0$) for the inflow boundary and the Neumann boundary conditions ($\partial \mathbf{u}/\partial n = 0$) for the other walls.

Fig. 15 presents the comparison of the transverse displacement of the midpoint A on the trailing edge between the present result and the reference data [21,40,41], where a good agreement can be observed. Moreover, comparisons of the quantitative results,

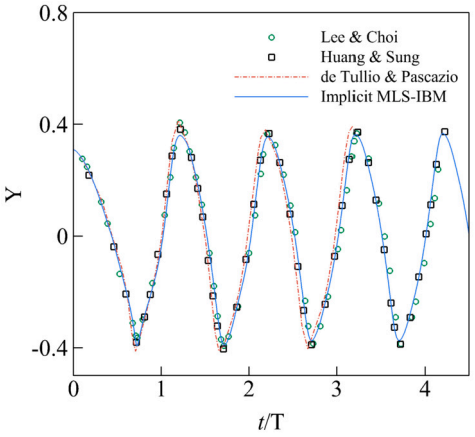

Fig. 15. Comparison of the transverse displacement of the midpoint A on the trailing edge.

Table 6
Comparisons of the flapping amplitude A/L and the Strouhal number $St = f_c L/U_0$, in which f_c denotes the flapping frequency.

$Re = 200$	Amplitude A/L	Strouhal number St
de Tullio & Pascazio [21]	0.795	0.265
Tian et al. - Flag 2 [44]	0.806	0.266
Huang & Sung [41]	0.780	0.260
Lee & Choi [40]	0.752	0.265
Implicit MLS-IBM	0.748	0.260

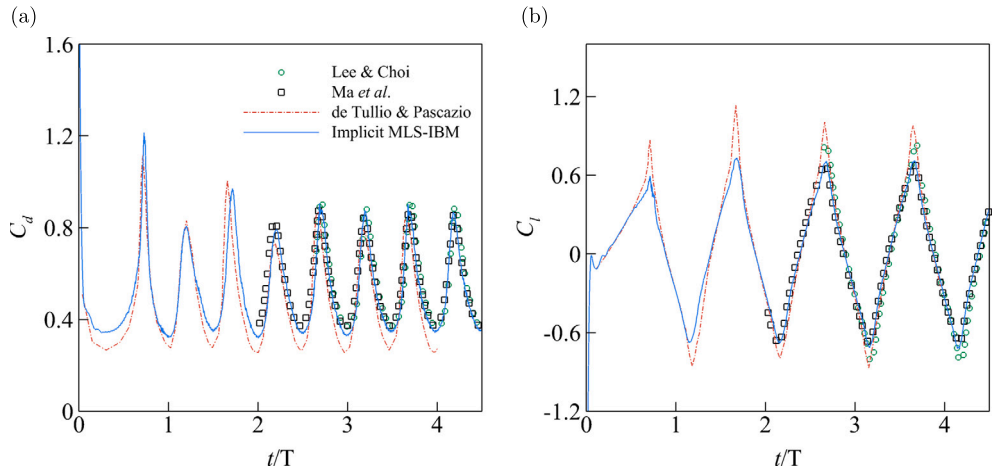

Fig. 16. Comparisons of time histories of the (a) drag and (b) lift coefficients.

including the flapping amplitude and the Strouhal number, are shown in Table 6. It is evident from Table 6 that the proposed implicit MLS-IBM can accurately predict the nonlinear characteristics of this complex FSI problems.

To further evaluate the accuracy of the aerodynamic forces predicted by the proposed implicit MLS-IBM, comparisons of the drag and lift coefficients between the present results and reference data are presented in Fig. 16, where the drag and lift coefficients are defined as $C_d = 2F_x/\rho U_0^2 L^2$ and $C_l = 2F_y/\rho U_0^2 L^2$, respectively. It can be clearly observed that the predicted aerodynamic forces are consistent with those from previous studies [21,40,43]. These quantitatively results demonstrate that the implicit MLS-IBM is capable of simulating challenging FSI problems with moving boundaries and large deformations.

Fig. 17 shows the evolution of vortical structures around the 3D flexible flag. It can be seen that as the 3D flag deforms, a longitudinal traveling wave can be clearly observed. In addition, a hairpin-like vortical structure is formed in the wake region, which is consistent with the observations from previous studies [21,40,41].

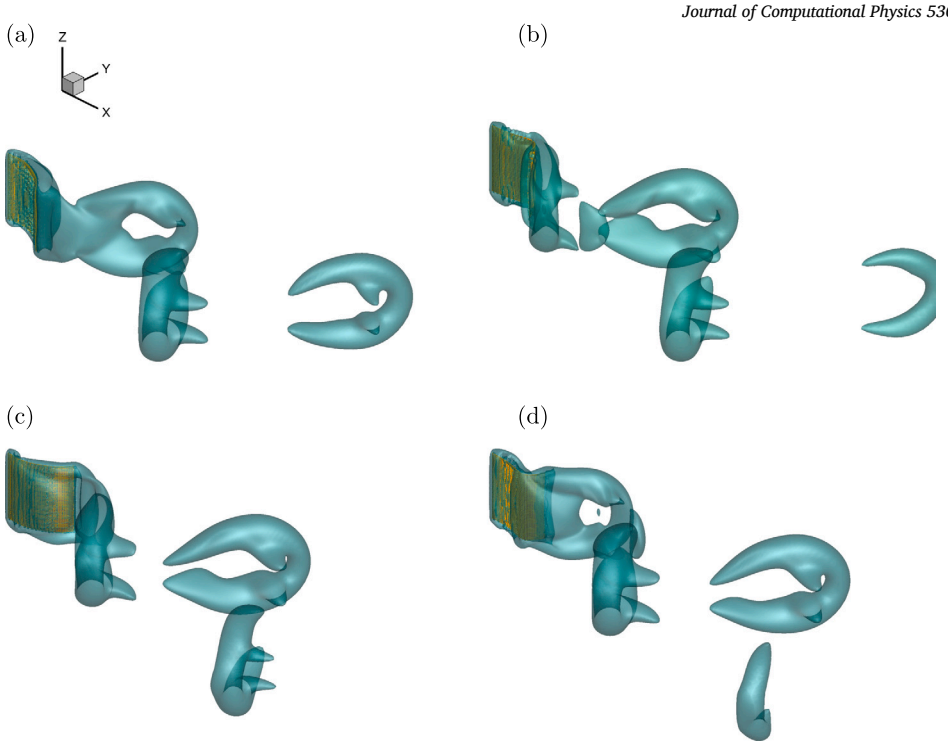

Fig. 17. The evolution of the instantaneous vortical structures around the 3D flexible flag.

6. Conclusions

In the present work, the implicit MLS-IBM is proposed, which not only eliminates the errors induced by the inequality between the interpolation and spreading operations but also preserves the conservation of force and torque. Furthermore, the construction of the correlation matrix containing the information of Lagrangian-Eulerian points is optimized to significantly promote the computational efficiency of the implicit MLS-IBM. In terms of errors in the velocity boundary condition and the conservation of force and torque, the proposed implicit MLS-IBM is much more accurate than the previous MLS-IBMs, i.e., the errors in the implicit MLS-IBM are reduced to the level of machine round-off. For stationary problems, the computational efficiency of the implicit MLS-IBM is more efficient than that of the explicit variant MLS-IBM. For moving problems, the computational efficiency of the implicit MLS-IBM is comparable to that of the explicit variant MLS-IBM and is substantially more efficient than conventional implicit IBMs, such as the implicit velocity correction IBM. These advantages indicate that the implicit MLS-IBM is suitable to simulate practical FSI problems. Nonetheless, when simulating complex problems with a large number of Lagrangian points, the whole computational procedure should be parallelized to accelerate the computational speed. Specifically, the proposed implicit system can be efficiently solved using the parallel GMRES solver on GPU platforms [32].

The proposed implicit MLS-IBM integrated with the flow solver (LBFS) has been validated with several complex FSI problems, including the self-propulsion of a 2D unconstrained flapping foil, the flow past a 3D torus and the flow past a 3D flexible flag. The global accuracy test demonstrates that the implicit MLS-IBM can achieve the second-order spatial accuracy. The numerical results provided by the implicit MLS-IBM demonstrate that the flow characteristics behind complex FSI problems are accurately captured and consistent with previous studies, qualitatively illustrating that the implicit MLS-IBM is capable of resolving the flow physics around immersed objects. Moreover, for both steady and unsteady flow problems, quantitative results predicted by the implicit MLS-IBM show good agreement with reference data. These numerical results demonstrate the capability and accuracy of the implicit MLS-IBM in simulating complex practical FSI problems with complex geometries, moving boundaries and large deformations.

CRediT authorship contribution statement

Buchen Wu: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Lin Fu:** Writing – review & editing, Writing – original draft, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Lin Fu acknowledges the fund from the National Natural Science Foundation of China (No. 12422210), the Research Grants Council (RGC) of the Government of Hong Kong Special Administrative Region (HKSAR) with RGC/ECS Project (No. 26200222), RGC/GRF Project (No. 16201023), RGC/GRF Project (No. 16206321), and RGC/STG Project (No. STG2/E-605/23-N), and the fund from Guangdong Basic and Applied Basic Research Foundation (No. 2024A1515011798).

Appendix A. Fluid-structure interaction approach

In this study, the FSI coupling procedure is resolved by a weak coupling approach, in which the flow behaviors and the solid motions are calculated sequentially within each time iteration [31,45]. The computational procedure of the FSI coupling is summarized as follows

(1) Initializing the flow field, the position and velocity of the immersed object, and the fluid force exerted on the immersed object is set to zero $\mathbf{F}_f = 0$;
(2) Solving the governing equation of solid motion to update the boundary position $\mathbf{X}^{n+1}$ and velocity $\boldsymbol{U}_b^{n+1}$;
(3) Utilizing the LBFS to predict the intermediate flow variables ρ^{n+1} and $\mathbf{u}^*$ through solving Eq. (2a);
(4) Using the proposed implicit MLS-IBM to evaluate the Lagrangian restoring force $\mathbf{F}_b^{n+1}$, then spread $\mathbf{F}_b^{n+1}$ to surrounding Eulerian points to get $\mathbf{f}^{n+1}$, and finally update the flow velocity $\mathbf{u}^{n+1}$;
(5) Calculating the fluid force exerted on the immersed object $\mathbf{F}_f = \mathbf{F}_b^{n+1}$;
(6) Repeating steps 2-5 until achieving final results.

Data availability

The data that support the findings of this study are available on request from the corresponding author, LF.

References

[1] R. Löhner, E. Haug, A. Michalski, B. Muhammad, A. Drego, R. Nanjundaiah, R. Zarfam, Recent advances in computational wind engineering and fluid–structure interaction, J. Wind Eng. Ind. Aerodyn. 144 (2015) 14–23.
[2] A. Khayyer, Y. Shimizu, H. Gotoh, S. Hattori, Multi-resolution isph-sph for accurate and efficient simulation of hydroelastic fluid-structure interactions in ocean engineering, Ocean Eng. 226 (2021) 108652.
[3] W. Xiao, H. Zhang, K. Luo, C. Mao, J. Fan, Immersed boundary method for multiphase transport phenomena, Rev. Chem. Eng. 38 (2022) 363–405.
[4] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220–252.
[5] E. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, J. Comput. Phys. 161 (2000) 35–60.
[6] Z.-G. Feng, E.E. Michaelides, Proteus: a direct forcing method in the simulations of particulate flows, J. Comput. Phys. 202 (2005) 20–51.
[7] M. Uhlmann, An immersed boundary method with direct forcing for the simulation of particulate flows, J. Comput. Phys. 209 (2005) 448–476.
[8] M. Chern, G. Lu, Y. Kuan, S. Chakraborty, G. Nugroho, C. Liao, T. Horng, Numerial study of vortex-induced vibration of circular cylinder adjacent to plane boundary using direct-forcing immersed boundary method, J. Mech. 34 (2018) 177–191.
[9] M. Uhlmann, Interface-resolved direct numerical simulation of vertical particulate channel flow in the turbulent regime, Phys. Fluids 20 (2008).
[10] A.G. Kidanemariam, M. Uhlmann, Direct numerical simulation of pattern formation in subaqueous sediment, J. Fluid Mech. 750 (2014) R2.
[11] T. Kempe, J. Fröhlich, An improved immersed boundary method with direct forcing for the simulation of particle laden flows, J. Comput. Phys. 231 (2012) 3663–3684.
[12] S. Wang, X. Zhang, An immersed boundary method based on discrete stream function formulation for two- and three-dimensional incompressible flows, J. Comput. Phys. 230 (2011) 3479–3499.
[13] K. Luo, Z. Wang, J. Fan, K. Cen, Full-scale solutions to particle-laden flows: multidirect forcing and immersed boundary method, Phys. Rev. E 76 (2007) 066709.
[14] W. Chen, S. Zou, Q. Cai, Y. Yang, An explicit and non-iterative moving-least-squares immersed-boundary method with low boundary velocity error, J. Comput. Phys. 474 (2023) 111803.
[15] J. Wu, C. Shu, Implicit velocity correction-based immersed boundary-lattice Boltzmann method and its applications, J. Comput. Phys. 228 (2009) 1963–1979.
[16] J. Favier, A. Revell, A. Pinelli, A lattice Boltzmann–immersed boundary method to simulate the fluid interaction with moving and slender flexible objects, J. Comput. Phys. 261 (2014) 145–161.
[17] M. Jiang, Z. Liu, A boundary thickening-based direct forcing immersed boundary method for fully resolved simulation of particle-laden flows, J. Comput. Phys. 390 (2019) 203–231.
[18] S.J. Shin, W.-X. Huang, H.J. Sung, Assessment of regularized delta functions and feedback forcing schemes for an immersed boundary method, Int. J. Numer. Methods Fluids 58 (2008) 263–286.
[19] X. Yang, X. Zhang, Z. Li, G.-W. He, A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations, J. Comput. Phys. 228 (2009) 7821–7836.
[20] M. Vanella, E. Balaras, A moving-least-squares reconstruction for embedded-boundary formulations, J. Comput. Phys. 228 (2009) 6617–6628.
[21] M.D. de Tullio, G. Pascazio, A moving-least-squares immersed boundary method for simulating the fluid–structure interaction of elastic bodies with arbitrary thickness, J. Comput. Phys. 325 (2016) 201–225.
[22] V. Spandan, V. Meschini, R. Ostilla-Mónico, D. Lohse, G. Querzoli, M.D. de Tullio, R. Verzicco, A parallel interaction potential approach coupled with the immersed boundary method for fully resolved simulations of deformable interfaces and membranes, J. Comput. Phys. 348 (2017) 567–590.
[23] A. Coclite, S. Ranaldo, M. De Tullio, P. Decuzzi, G. Pascazio, Kinematic and dynamic forcing strategies for predicting the transport of inertial capsules via a combined lattice Boltzmann–immersed boundary method, Comput. Fluids 180 (2019) 41–53.
[24] Y. Wang, C. Shu, L. Yang, Boundary condition-enforced immersed boundary-lattice Boltzmann flux solver for thermal flows with Neumann boundary conditions, J. Comput. Phys. 306 (2016) 237–252.
[25] J. Lu, H. Lei, C. Dai, L. Yang, C. Shu, Analyses and reconstruction of the lattice Boltzmann flux solver, J. Comput. Phys. (2022) 110923.

[26] S. Succi, The Lattice Boltzmann Equation for Fluid Dynamics and Beyond, Oxford University Press, 2001.
[27] J. Wu, C. Shu, An improved immersed boundary-lattice Boltzmann method for simulating three-dimensional incompressible flows, J. Comput. Phys. 229 (2010) 5022–5042.
[28] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.
[29] Z. Li, G. Oger, D. Le Touzé, A partitioned framework for coupling lbm and fem through an implicit ibm allowing non-conforming time-steps: application to fluid-structure interaction in biomechanics, J. Comput. Phys. 449 (2022) 110786.
[30] X. Zhao, Z. Chen, L. Yang, N. Liu, C. Shu, Efficient boundary condition-enforced immersed boundary method for incompressible flows with moving boundaries, J. Comput. Phys. (2021) 110425.
[31] Y. Wang, C. Shu, T. Wang, P. Valdivia y Alvarado, A generalized minimal residual method-based immersed boundary-lattice Boltzmann flux solver coupled with finite element method for non-linear fluid-structure interaction problems, Phys. Fluids 31 (2019).
[32] K. He, S.X.-D. Tan, H. Zhao, X.-X. Liu, H. Wang, G. Shi, Parallel gmres solver for fast analysis of large linear dynamic systems on gpu platforms, Integration 52 (2016) 10–22.
[33] T. Xu, J.-I. Choi, Efficient monolithic immersed boundary projection method for incompressible flows with heat transfer, J. Comput. Phys. 477 (2023) 111929.
[34] Q. Zhou, L.-S. Fan, A second-order accurate immersed boundary-lattice Boltzmann method for particle-laden flows, J. Comput. Phys. 268 (2014) 269–301.
[35] J. Lu, H. Lei, C. Shu, C. Dai, The more actual macroscopic equations recovered from lattice Boltzmann equation and their applications, J. Comput. Phys. 415 (2020) 109546.
[36] A.J. Ladd, R. Verberg, Lattice-Boltzmann simulations of particle-fluid suspensions, J. Stat. Phys. 104 (2001) 1191–1251.
[37] X. Lin, J. Wu, T. Zhang, Self-directed propulsion of an unconstrained flapping swimmer at low Reynolds number: hydrodynamic behaviour and scaling laws, J. Fluid Mech. 907 (2021).
[38] Y. Wang, C. Shu, C. Teo, L. Yang, An efficient immersed boundary-lattice Boltzmann flux solver for simulation of 3d incompressible flows with complex geometry, Comput. Fluids 124 (2016) 54–66.
[39] L. Yang, C. Shu, W. Yang, Y. Wang, J. Wu, An immersed boundary-simplified sphere function-based gas kinetic scheme for simulation of 3d incompressible flows, Phys. Fluids 29 (2017) 083605.
[40] I. Lee, H. Choi, A discrete-forcing immersed boundary method for the fluid–structure interaction of an elastic slender body, J. Comput. Phys. 280 (2015) 529–546.
[41] W.-X. Huang, H.J. Sung, Three-dimensional simulation of a flapping flag in a uniform flow, J. Fluid Mech. 653 (2010) 301.
[42] T. Gao, L. Fu, A three-dimensional meshless fluid–shell interaction framework based on smoothed particle hydrodynamics coupled with semi-meshless thin shell, Comput. Methods Appl. Mech. Eng. 429 (2024) 117179.
[43] J. Ma, Z. Wang, J. Young, J.C. Lai, Y. Sui, F.-B. Tian, An immersed boundary-lattice Boltzmann method for fluid-structure interaction problems involving viscoelastic fluids and complex geometries, J. Comput. Phys. 415 (2020) 109487.
[44] F.-B. Tian, H. Dai, H. Luo, J.F. Doyle, B. Rousseau, Fluid–structure interaction involving large deformations: 3d simulations and applications to biological systems, J. Comput. Phys. 258 (2014) 451–469.
[45] L. Wang, G.M. Currao, F. Han, A.J. Neely, J. Young, F.-B. Tian, An immersed boundary method for fluid–structure interaction with compressible multiphase flows, J. Comput. Phys. 346 (2017) 131–151.