On Advances of Diffuse-Interface Immersed Boundary Method and Its Applications

YANG Liming[1], *SHU Chang*[1,2*], *DU Yinjie*[1], *WU Jie*[1], *WANG Yan*[1]

1. College of Aerospace Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, P. R. China;
2. Department of Mechanical Engineering, National University of Singapore, Singapore 117576, Singapore

(Received 28 February 2025; revised 21 March 2025; accepted 7 April 2025)

Abstract: The diffuse-interface immersed boundary method (IBM) possesses excellent capabilities for simulating flows around complex geometries and moving boundaries. In this method, the flow field is solved on a fixed Cartesian mesh, while the solid boundary is discretized into a series of Lagrangian points immersed in the flow field. The boundary condition is implemented by introducing a force term into the momentum equation, and the interaction between the immersed boundary and the fluid domain is achieved via an interpolation process. Over the past decades, the diffuse-interface IBM has gained popularity and spawned many variants, effectively handling a wide range of flow problems from isothermal to thermal flows, from laminar to turbulent flows, and from complex geometries to fluid-structure interaction scenarios. This paper first outlines the basic principles of the diffuse-interface IBM, then highlights recent advancements achieved by the authors' research group, and finally shows the method's excellent numerical performance and wide applicability through several case studies involving complex moving boundary problems.

Key words: immersed boundary method; diffuse-interface; moving boundary; incompressible flows; turbulent flows

CLC number: O354 **Document code:** A **Article ID:** 1005-1120(2025)02-0137-25

0 Introduction

In addition to theoretical analysis and experiments, numerical simulation, commonly known as computational fluid dynamics (CFD), is an essential method for studying fluid flow problems, forming a significant branch of fluid mechanics. Nowadays, CFD has been extensively used in aerospace, meteorology, automotive, shipbuilding, and oceanography, leading to notable achievements. The fundamental strategy of CFD is to discretize the fluid domain using a grid and to numerically solve a set of partial differential equations on that grid. Therefore, the quality of the grid is crucial, as it directly affects the accuracy and efficiency of the numerical simulations.

Conventional CFD methods typically adopt body-fitted grids, where the grid lines conform to the solid boundaries, simplifying the imposition of boundary conditions. However, in practical applications, the solid geometries are usually complex, making the generation of body-fitted grids difficult and cumbersome. Moreover, moving boundary problems necessitate regenerating body-fitted grids at every time step, which substantially increases computational cost and reduces efficiency[1]. In contrast, the immersed boundary method (IBM)[2-6] can handle complex geometries and moving boundary problems on a fixed Cartesian grid, attracting widespread attention and rapid development in recent decades. The IBM was originally proposed by Peskin[7] in 1972 to simulate the blood flow around heart valves. In his approach, the flow field is solved on a fixed Cartesian (Eulerian) mesh and the

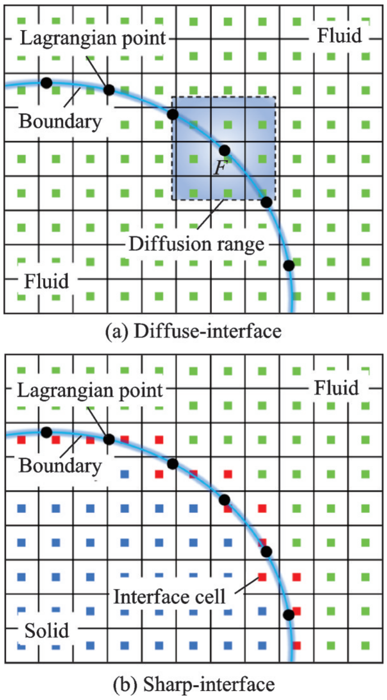
*Corresponding author, E-mail address: mpeshuc@nus.edu.sg.

How to cite this article: YANG Liming, SHU Chang, DU Yinjie, et al. On advances of diffuse-interface immersed boundary method and its applications[J]. Transactions of Nanjing University of Aeronautics and Astronautics, 2025,42(2):137-161.

http://dx.doi.org/10.16356/j.1005-1120.2025.02.001

solid boundary is represented by a series of discrete (Lagrangian) points immersed in the flow field. The influence of the immersed boundary on the flow is incorporated by adding a force term to the momentum equation. This approach decouples the mesh generation from the boundary geometry, avoiding the cumbersome and time-consuming mesh generation process encountered by conventional CFD methods when dealing with complex geometries and moving boundary problems. Since then, the approach of the "immersed boundary" has been extensively developed, leading to numerous variants of the IBM.

Depending on how boundary conditions are treated, existing IBMs can generally be categorized into diffuse-interface and sharp-interface approaches, as illustrated in Fig.1. The original IBM proposed by Peskin[7] is a typical diffuse-interface IBM, in which the force term $\boldsymbol{F}$ is evaluated at the Lagrangian points and then distributed to the surrounding fluid points via a smooth interpolation function, ensuring a gradual transition from the boundary to the fluid domain. In this approach, all mesh cells both inside and outside the boundary are treated uniformly as fluid cells, as shown in Fig.1(a). In contrast, the sharp-interface IBM[8-10] divides the mesh cells into interface cells, fluid cells, and solid cells, as illustrated in Fig.1(b). This method enforces the boundary condition by directly reconstructing the velocity in the interface (or ghost) cells, which are interpolated from the surrounding fluid cells and the boundary. Common approaches include the cut-cell method[11-12], the ghost-cell method[13-14], and the ghost-fluid method[15-16], in which the immersed boundary is preserved as a "sharp" interface without smearing. However, distinguishing the cells in the sharp-interface IBM is both tedious and complex, and due to discontinuities in the mesh cells near the immersed boundary, this method can suffer from spurious oscillations, particularly in moving boundary problems.



(a) Diffuse-interface

(b) Sharp-interface

Fig.1 Comparison of the diffuse-interface IBM and the sharp-interface IBM

The diffuse-interface IBM originally proposed by Peskin[7] treated the boundary as elastic and applied Hooke' s law to calculate the force term, a technique known as the penalty forcing method. Since its introduction, significant progress has been made in developing the diffuse-interface IBM. To ensure smooth transmission of the force term, Peskin' s group[2, 17-19] developed smooth Dirac delta functions to model the interaction between the fluid and the immersed boundary. These include the 3-point discrete piecewise function[18], the 4-point cosine function[17], and the 4-point discrete piecewise function[19], which have been widely adopted in various diffuse-interface IBMs. However, because the penalty forcing method relies on Hooke's law to calculate the force term, it is more suitable for elastic bodies than for rigid ones. To simulate flows around rigid bodies, Goldstein et al.[20] proposed a feedback forcing method, which involves integrating the boundary velocity. Both the penalty forcing and feedback forcing methods depend on ad hoc coefficients, which can hinder their broader adoption. Fadlun et al.[21] then developed a direct forcing method, in which the desired velocity is imposed directly on the boundary without any dynamic process or user-defined coefficients. Note, however, that the original direct forcing method[21] is not a typical diffuse-interface IBM, as it calculates the force term directly at the Eulerian cells near the boundary by approximating the desired velocity using linear interpolation. Lima E Silva et al.[22] proposed a physical virtual model to eliminate the need for such ad hoc constants by applying the Navier-Stokes (N-S) equations at the Lagrangian points to evaluate the force term. However, this method requires tedious derivative approximations and interpolations of the pressure and velocity. A simpler and more widely used version was later developed by Uhlmann[23], who improved the direct forcing method by incorporating a smooth Dirac delta function and the fractional-step technique. This improved scheme directly evaluates the force term based on the difference between the desired and predicted velocities at the Lagrangian point and then transfers it to the Eulerian mesh through the smooth Dirac delta function. Due to the common use of Cartesian meshes in both the IBM and the lattice Boltzmann method (LBM), some researchers integrated IBM into LBM to develop efficient algorithms. For example, by adding a force term to the collision term of the fundamental LBM equation, Feng and Michaelides[24-25] successfully applied both the penalty forcing scheme[7] and the direct forcing scheme[23] to simulate particulate flows within the LBM framework. Additionally, since the density distribution function serves as the evolution variable in LBM, Niu et al.[26] proposed a momentum exchange-based IBM, in which the force term is evaluated through the momentum exchange of the boundary particle density distribution functions.

Examining the evolution of the diffuse-interface IBM reveals that the calculation of the force term is essential to its performance. However, a persistent drawback is that the explicitly calculated force term cannot accurately enforce the no-slip boundary condition, resulting in non-physical phenomena such as streamlines penetrating the boundaries at convergence. To address this issue, Luo et al.[27] and Wang et al.[28] developed a multi-direct forcing method that iteratively used the direct forcing scheme to ensure that the fluid velocity at the immersed boundary closely approximates the desired boundary velocity. Hu et al.[29] improved the momentum exchange-based IBM by introducing an iterative technique to enforce the no-slip condition. Chen et al.[30] attempted to reduce the boundary velocity error in the direct forcing method by introducing a spatially uniform coefficient to correct the calculated force term. Shu et al.[31] observed that the conventional diffuse-interface IBM's inability to accurately satisfy the no-slip boundary condition stems from the pre-calculation of the force term. Using a fractional-step technique, they concluded that adding a force term in the momentum equations was equivalent to correcting the velocity field. Consequently, they proposed an immersed boundary velocity correction method that enforced the no-slip condition by adjusting velocities at points adjacent to the boundary along horizontal and vertical mesh lines. Building on this concept, Wu and Shu[32-33] developed an implicit velocity correction-based IBM, also known as the boundary condition-enforced IBM, in which the velocity correction (i.e., the force term) was treated as an unknown and was implicitly determined by enforcing the no-slip condition. This method couples the velocity corrections at all Lagrangian points into a matrix system for computation. Although the boundary condition-enforced IBM accurately enforces the no-slip condition and effectively eliminates non-physical penetration of streamlines, solving the matrix system is computationally expensive, as its size depends on the number of Lagrangian points.

To mitigate the computational burden of solving the matrix system, Dash et al.[34-35] proposed a flexible forcing IBM that merges the concepts of implicit velocity correction[32-33] and multi-direct forcing[27-28]. This approach introduces a sub-iteration update scheme with a convergence criterion to satisfy the no-slip condition. Based on the Taylor series analysis, Zhao et al.[36] developed an explicit version of the boundary condition-enforced IBM, improving efficiency by avoiding time-consuming matrix operations. These two improved schemes are indeed approximate solutions to the velocity correction matrix system. Furthermore, to overcome the limitation of the original boundary condition-enforced IBM, where the solid body must be immersed in a uniform mesh region due to the use of the smooth Dirac delta function, Du et al.[37] introduced the inverse distance weighting (IDW) interpolation[38-39] to associate Lagrangian points with surrounding Eulerian points. This extension broadens the applicability of the boundary condition-enforced IBM to non-uniform meshes while reducing the required number of Lagrangian points. Additionally, Du et al.[40] developed a virtual body-fitted grid-based IBM, which enforced the no-slip condition on a local virtual grid. This virtual grid decouples the spacing of Lagrangian points from that of the Eulerian mesh, allowing for a more flexible distribution of both Lagrangian points and Eulerian cells. Due to their simplicity and robustness, the boundary condition-enforced IBM[32-33] and its variants[34-37, 40-41] have been extensively used to simulate incompressible flows involving complex geometries and moving boundaries.

In addition to incompressible isothermal flows, the applications of the diffuse-interface IBM have been extended to thermal and multiphase flows. Thermal flow simulations must account for temperature boundary conditions, which typically include the isothermal condition and the iso-heat-flux condition. The isothermal condition, like the no-slip condition, is a Dirichlet-type condition. Therefore, by incorporating a heating term into the energy equation, the isothermal boundary condition can be enforced in the same manner as in the no-slip condition. This approach has been successfully used by Zhang et al.[42] and Ren et al.[43] to extend their IBM for thermal flows. In contrast, the iso-heat-flux condition is a Neumann-type condition that cannot be directly handled by conventional diffuse-interface IBMs. To address this, Zhang et al.[42] converted the boundary heat flux into a boundary temperature using a difference approximation based on the temperature of an auxiliary layer, while Ren et al.[44] developed a heat flux correction-based IBM that directly adjusted the temperature field utilizing the offset of the boundary temperature derivative. Additionally, Wang et al.[45] introduced two auxiliary layers of Lagrangian points, located inside and outside the solid body, and used the temperature difference between these layers to approximate the iso-heat-flux condition. Moreover, the virtual body-fitted grid-based method proposed by Du et al.[46] can easily handle the iso-heat-flux condition, as the local virtual grid extends in the wall-normal direction. Temperatures at virtual layers inside the immersed body are corrected using those from virtual layers outside the wall following a quadratic distribution, effectively satisfying the iso-heat-flux condition. For multiphase flow simulations, Shao et al.[47] designed an IBM-based method for implementing Neumann boundary conditions within the phase-field LBM and used it to study contact line dynamics. Wang et al.[48] developed a two-dimensional IBM for fluid-structure interactions with compressible multiphase flows, incorporating large structural deformations. Their approach employs a partitioned iterative coupling strategy with a feedback penalty IBM. Furthermore, Huang et al. [49] proposed an improved version of the penalty IBM for simulating multiphase flows with nonuniform density and a moving interface.

The aforementioned developments of the diffuse-interface IBM primarily focus on simulating laminar flows at low and moderate Reynolds numbers. Turbulent flows at high Reynolds numbers pose a significant challenge to the diffuse-interface IBM due to the presence of a thin boundary layer. In the literature, various strategies have been developed for the sharp-interface approach, where velocity at the interface cell is reconstructed directly using wall models[45-55]. In contrast, the smooth interpolation function used in the diffuse-interface approach to mediate interactions between the immersed boundary and the fluid field complicates the integration of wall models. Recently, several studies have attempted to address this issue. Based on the direct forcing IBM, Shi et al.[56] integrated the tangential momentum equation along the wall-normal direction, linking the force term with the wall shear stress obtained from a wall model. This method was later incorporated into the boundary condition-enforced IBM[32-33] and further refined by Du et al.[57], who improved the tangential force term after integration to enhance computational accuracy. Additionally, Ma et al.[58-59] used the penalty forcing method[60] to enforce the no-slip boundary condition in their large eddy simulations, while correcting the subgridscale viscosity of Eulerian cells near the boundary based on a wall model. Yan et al.[61] employed a wall model to obtain a tangential slip velocity as a boundary condition, which was then enforced using the explicit boundary condition-enforced IBM[36]. Despite these advancements, the simulation of turbulent flows at high Reynolds numbers using the diffuse-interface IBM still requires further improvement and exploration. Typically, a wall model is used at a reference point near the boundary to compute the wall shear stress. For greater accuracy, this reference point should be as close to the boundary as possible. However, in the diffuse-interface IBM, the reference point must remain outside the interpolation range of the smooth function to avoid computational instability. Currently, this contradiction can be alleviated by refining the Eulerian mesh, but doing so reduces computational efficiency.

The discussion above provides a brief review of the development and progress of the diffuse-interface IBM. Overall, the diffuse-interface IBM is a highly promising method for simulating flows with complex geometries and moving boundaries. In the following sections, we will present its algorithm and applications in detail.

1 Governing Equations

Unlike conventional body-fitted mesh methods, the diffuse-interface IBM introduces a force term into the momentum equation to account for the influence of the boundary on the flow field. This section provides a brief overview of the governing equations for incompressible flows in the diffuse-interface IBM.

1.1 Governing equations for incompressible isothermal flows

For incompressible isothermal flows, the governing equations consist of the continuity equation and the momentum equation, expressed as

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

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u} + \boldsymbol{f} \tag{1b}$$

where ρ, p, t, and ν denote the density, pressure, time, and kinematic viscosity, respectively; $\boldsymbol{u} = [u \quad v \quad w]^{\mathrm{T}}$ represents the velocity vector and $\boldsymbol{f} = [f_x \quad f_y \quad f_z]^{\mathrm{T}}$ the force term exerted by the immersed boundary on the fluid.

1.2 Governing equations for incompressible thermal flows

For incompressible thermal flows, the energy equation is included to account for heat transfer effects, incorporating a heat source term to represent the influence of a heated immersed boundary. The governing equations are given by

$$\nabla \cdot \boldsymbol{u} = 0 \tag{2a}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u} + \boldsymbol{F}_{\mathrm{b}} + \boldsymbol{f} \tag{2b}$$

$$\frac{\partial T}{\partial t} + \boldsymbol{u} \cdot \nabla T = \kappa \nabla^2 T + q \tag{2c}$$

where T is the temperature, κ the thermal diffusivity coefficient, and q the heat source term. Additionally; $\boldsymbol{F}_{\mathrm{b}}$ represents the buoyancy force based on the Boussinesq approximation, which is a key external force in natural or mixed convection problems. It is defined as

$$\boldsymbol{F}_{\mathrm{b}} = [0 \quad -g\beta(T - T_{\mathrm{ref}}) \quad 0]^{\mathrm{T}} \tag{3}$$

where g represents the gravitational acceleration, β the thermal expansion coefficient, and T_{ref} the reference temperature. Here, the buoyancy force acts only in the y-direction.

1.3 Governing equations for incompressible turbulent flows

For incompressible turbulent flows, the incompressible Reynolds-Averaged Navier-Stokes (RANS) equations involving a force term can be expressed as

$$\nabla \cdot \bar{\boldsymbol{u}} = 0 \tag{4a}$$

$$\frac{\partial \bar{\boldsymbol{u}}}{\partial t} + \bar{\boldsymbol{u}} \cdot \nabla \bar{\boldsymbol{u}} = -\frac{1}{\rho}\nabla \bar{p} + \nabla \cdot ((\nu + \nu_t)\nabla \bar{\boldsymbol{u}}) + \boldsymbol{f} \tag{4b}$$

where the superscript "–" denotes the Reynolds-Averaged variables; and ν_t represents the eddy viscosity, which can be calculated by turbulence models, such as the Spalart-Allmaras (S-A) model[62] and the k-ω shear stress transport (SST) model[63]. Alternatively, the large eddy simulation (LES) is another widely used approach for simulating turbulent flows and has also been combined with IBM in various studies. The governing equations for the LES are the filtered N-S equations, which take the same form as Eq.(4). In this case, the superscript "–" represents the filtered variables, and the eddy viscosity ν_t can be determined by the Smagorinsky eddy viscosity model[64].

2 Diffuse-Interface Immersed Boundary Method

In the diffuse-interface IBM, the flow field is solved on a Cartesian (Eulerian) mesh, while the solid boundary is represented by a set of discrete (Lagrangian) points immersed in the flow field. This section introduces the fundamental principles of the diffuse-interface IBM, followed by a focus on the boundary condition-enforced diffuse-interface IBM and its variants.

2.1 Basic principles of the diffuse-interface IBM

The diffuse-interface IBM evaluates the force terms at the Lagrangian points and distributes them to the surrounding Eulerian cells. Generally, the governing equations incorporating the force term can be solved using two different approaches, leading to two classifications of the diffuse-interface IBM. One is the discrete forcing approach[3], which used the fractional-step technique, comprising a predictor step and a corrector step, to solve the governing equations. The prediction step solves the governing equations without the force term to obtain an intermediate flow field on the Eulerian mesh. Several solvers can be used for this step, including the projection method[65], the SIMPLE method[66], the LBM[32], and the finite volume-lattice Boltzmann flux solver[67]. The corrector step then modifies the intermediate flow field by incorporating the force term. In this approach, the force term is introduced after discretizing the equations. Unlike the discrete forcing method, the other method incorporates the force term into the governing equations before discretization, which is called the continuous forcing approach[3].

A notable example of the continuous forcing approach is the penalty forcing method, proposed by Peskin[7], which uses Hooke's law to compute the force term

$$\boldsymbol{F}_m = -k(\boldsymbol{X}_m - \boldsymbol{X}_m^{\mathrm{e}}) \tag{5}$$

where $\boldsymbol{F}_m$ is the force at the mth Lagrangian point and k a positive spring constant. $\boldsymbol{X}_m$ and $\boldsymbol{X}_m^{\mathrm{e}}$ denote the actual and equilibrium positions of the mth Lagrangian point, respectively. This method is particularly suitable for elastic bodies. Goldstein et al.[20] extended this approach by developing a feedback-forcing method for rigid bodies, expressed as

$$\boldsymbol{F}_m = \alpha\int_0^t \boldsymbol{U}_m(t')\,\mathrm{d}t' + \beta\boldsymbol{U}_m(t) \tag{6}$$

where $\boldsymbol{U}_m$ is the velocity at the mth Lagrangian point, and α and β are the negative constants. The penalty forcing method[7] can be viewed as a specific case of this model. However, both approaches involve artificial coefficients, limiting their broader applicability.

A more commonly used approach is the direct forcing method, introduced by Uhlmann[23]. This method determines the force term based on the difference between the predicted velocity $\boldsymbol{U}_m^*$ and the desired velocity $\boldsymbol{U}_m^{\mathrm{B}}$ at the Lagrangian point, as follows

$$\boldsymbol{F}_m = \frac{\boldsymbol{U}_m^{\mathrm{B}} - \boldsymbol{U}_m^*}{\Delta t} \tag{7}$$

This method is widely used due to its simplicity and the absence of artificial constants.

Besides, Niu et al.[26] introduced a momentum exchange-based method for the LBM, where the force term was evaluated based on the density distribution function

$$\boldsymbol{F}_m = \sum_j \boldsymbol{e}_j(f_{m,j} - f_{m,i}) \tag{8}$$

where f and $\boldsymbol{e}$ are the density distribution function and the lattice velocity, respectively; i and j represent the lattice velocity directions which are opposite to each other.

As illustrated in Fig.2, the computed force term is then distributed to the surrounding Eulerian cells using an interpolation function, typically a smooth Dirac delta function[17-19]

$$\boldsymbol{f}_n=\sum_m \boldsymbol{F}_m \Delta s_m D(\boldsymbol{x}_n-\boldsymbol{X}_m) \tag{9}$$

where Δs_m is the area of the mth boundary element; $\boldsymbol{x}_n$ is the position of the nth Eulerian cell; and $D(\boldsymbol{x}_n-\boldsymbol{X}_m)$ is composed of the one-dimensional discrete Dirac delta function, expressed as

$$D(\boldsymbol{x}_n-\boldsymbol{X}_m)=\frac{1}{h^3}\delta\left(\frac{x_n-X_m}{h}\right)\delta\left(\frac{y_n-Y_m}{h}\right)\delta\left(\frac{z_n-Z_m}{h}\right) \tag{10}$$

where h is the mesh spacing of the Eulerian mesh.

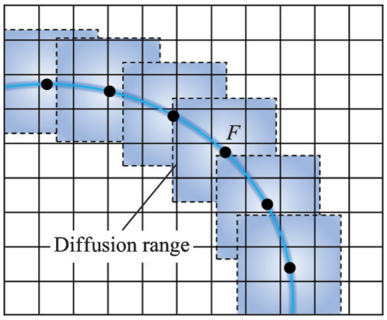


Fig.2 Influence ranges of Lagrangian points when using the smooth Dirac delta function for interpolation

Commonly used discrete delta functions include the 2-point hat function[68], shown as

$$\delta(r)=\begin{cases}1-|r| & |r|\leqslant 1\\ 0 & |r|>1\end{cases} \tag{11}$$

the 3-point discrete piecewise function[18], shown as

$$\delta(r)=\begin{cases}\frac{1}{3}\left(1+\sqrt{1-3|r|^2}\right) & |r|\leqslant 0.5\\ \frac{1}{6}\left(5-3|r|-\sqrt{1-3(1-|r|)^2}\right) & 0.5<|r|\leqslant 1.5\\ 0 & |r|>1.5\end{cases} \tag{12}$$

the 4-point cosine function[17], shown as

$$\delta(r)=\begin{cases}\frac{1}{4}\left(1+\cos\left(\frac{\pi|r|}{2}\right)\right) & |r|\leqslant 2\\ 0 & |r|>2\end{cases} \tag{13}$$

and the 4-point discrete piecewise function[19], shown as

$$\delta(r)=\begin{cases}\frac{1}{8}\left(3-2|r|+\sqrt{1+4|r|-4|r|^2}\right) & |r|\leqslant 1\\ \frac{1}{8}\left(5-2|r|-\sqrt{-7+12|r|-4|r|^2}\right) & 1<|r|\leqslant 2\\ 0 & |r|>2\end{cases} \tag{14}$$

In addition to the above, Yang et al.[69] proposed a smoothing technique to develop new smooth discrete delta functions. These functions, with derivatives satisfying a higher-order moment condition, effectively suppress non-physical oscillations in moving boundary simulations. However, all these smooth discrete delta functions are limited to uniform Cartesian meshes. To extend IBM to non-uniform meshes, Du et al.[37] introduced inverse distance weight interpolation as an alternative to the discrete delta function. Additionally, Vanella et al.[70] used moving least squares reconstruction to establish transfer functions between the Eulerian mesh and Lagrangian points.

While the methods discussed above explicitly calculate the force term, there also exist implicit IBMs, where the force is determined implicitly. For example, Taira et al.[65] proposed a fully implicit IBM using the projection method, treating both the force term and pressure as a single set of Lagrange multipliers in a modified Poisson equation. This method enforces both the no-slip condition and incompressibility constraint simultaneously through a projection. Similarly, Goncharuk et al.[66] developed an implicit direct forcing IBM based on the SIMPLE algorithm, which couples pressure, velocity, and force term corrections while enforcing incompressibility and the no-slip condition.

2.2 Boundary condition-enforced diffuse-interface IBM

In the conventional diffuse-interface IBM, the force term is evaluated locally and explicitly at each Lagrangian point. However, this approach fails to strictly enforce the no-slip boundary condition, leading to non-physical flow leakage through the immersed boundary. To address this issue, Shu et al.[31] proposed the immersed boundary velocity correction method (IBVCM), which directly corrected the velocities at points adjacent to the boundary using linear interpolation. This method is based on the observation that, in a fractional-step technique, adding a force term to the momentum equation is equivalent to correcting the velocity field. The correction step is given by

$$\frac{\partial \boldsymbol{u}}{\partial t}=\boldsymbol{f} \tag{15}$$

which can be discretized as

$$\boldsymbol{u}^{l+1}=\boldsymbol{u}^{*}+\Delta t\boldsymbol{f}=\boldsymbol{u}^{*}+\Delta \boldsymbol{u} \tag{16}$$

where the superscript "$l+1$" represents the next time step and "*" the predicted intermediate flow variables. Building on the IBVCM[31], Wang et al.[71] introduced the second-order Lagrange interpolation to enhance the accuracy of velocity correction. The IBVCM[31] corrects velocities at Eulerian points located on either side of the intersection points between mesh lines and the immersed boundary. This differs from the conventional diffuse-interface IBM and may introduce oscillations near the boundary.

Following the concept of velocity correction[31], Wu and Shu[32] developed the boundary condition-enforced IBM. This method ensures the no-slip boundary condition by enforcing the velocity at Lagrangian points, interpolated from surrounding Eulerian cells, to match the boundary velocity, shown as

$$\boldsymbol{U}_m^{\mathrm{B}}=\sum_n \boldsymbol{u}_n^{l+1}h^3D(\boldsymbol{X}_m-\boldsymbol{x}_n) \tag{17}$$

Here, a smooth Dirac delta function[17-19] is used to interpolate the velocities from the Eulerian cells to the Lagrangian points. Conversely, the velocity corrections (equivalent to the force term) for Eulerian cells are distributed from the velocity corrections of Lagrangian points, following the same procedure as in Eq.(9), i.e.

$$\Delta\boldsymbol{u}_n=\sum_m \Delta\boldsymbol{U}_m^{\mathrm{B}}\Delta s_m D(\boldsymbol{x}_n-\boldsymbol{X}_m) \tag{18}$$

Combining Eqs.(16—18) results in the following equation

$$\boldsymbol{U}_m^{\mathrm{B}}=\sum_n \boldsymbol{u}_n^{*}h^3D(\boldsymbol{X}_m-\boldsymbol{x}_n)+\sum_n h^3D(\boldsymbol{X}_m-\boldsymbol{x}_n)\sum_m \Delta\boldsymbol{U}_m^{\mathrm{B}}\Delta s_m D(\boldsymbol{x}_n-\boldsymbol{X}_m) \tag{19}$$

which can be rewritten in a matrix form as

$$\boldsymbol{AX}=\boldsymbol{B} \tag{20}$$

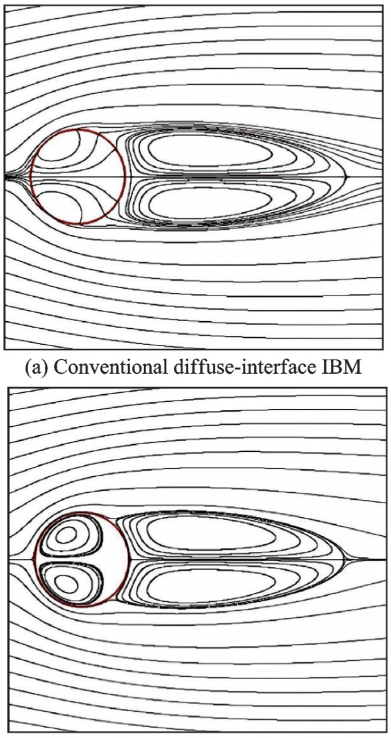
where

$$\boldsymbol{X}=[\Delta\boldsymbol{U}_1\Delta s_1 \quad \Delta\boldsymbol{U}_2\Delta s_2 \quad \cdots \quad \Delta\boldsymbol{U}_M\Delta s_M]^{\mathrm{T}}$$

$$\boldsymbol{A}=h^3\begin{bmatrix} D_{11} & D_{12} & \cdots & D_{1N} \\ D_{21} & D_{22} & \cdots & D_{2N} \\ \vdots & \vdots & & \vdots \\ D_{M1} & D_{M2} & \cdots & D_{MN} \end{bmatrix}\begin{bmatrix} D_{11} & D_{12} & \cdots & D_{1M} \\ D_{21} & D_{22} & \cdots & D_{2M} \\ \vdots & \vdots & & \vdots \\ D_{N1} & D_{N2} & \cdots & D_{NM} \end{bmatrix}$$

$$\boldsymbol{B}=\begin{bmatrix} \boldsymbol{U}_1^B \\ \boldsymbol{U}_2^B \\ \vdots \\ \boldsymbol{U}_M^B \end{bmatrix}-\begin{bmatrix} D_{11} & D_{12} & \cdots & D_{1N} \\ D_{21} & D_{22} & \cdots & D_{2N} \\ \vdots & \vdots & & \vdots \\ D_{M1} & D_{M2} & \cdots & D_{MN} \end{bmatrix}\begin{bmatrix} \boldsymbol{u}_1^* \\ \boldsymbol{u}_2^* \\ \vdots \\ \boldsymbol{u}_N^* \end{bmatrix}h^3$$

Here, M and N represent the number of Lagrangian points and affected Eulerian cells, respectively. By solving Eq.(20) and applying Eq.(16) and Eq.(18), the underlying flow field can be corrected. A key advantage of this method is that the term "$\Delta\boldsymbol{U}_m\Delta s_m$" is treated as an unknown, eliminating the need to evaluate the area Δs of the boundary element. In contrast, the conventional IBM approaches typically assume an immersed boundary thickness of h to calculate the area Δs. Throughout the process, the boundary condition-enforced IBM does not explicitly compute the force term. Instead, it implicitly couples the velocity corrections at all Lagrangian points, ensuring strict enforcement of the no-slip boundary condition and preventing non-physical streamline penetration through the boundary. This improvement is demonstrated by the comparison of streamlines in Fig.3[32] for flow around a circular cylinder at $Re=40$.



(a) Conventional diffuse-interface IBM

(b) Boundary condition-enforced IBM

Fig.3 Comparison of streamlines obtained by using different methods for the flow around a circular cylinder at $Re=40$[32]

To further improve computational efficiency, Zhao et al.[36] developed an explicit boundary condition-enforced IBM, eliminating time-consuming matrix operations. Eq.(20) can be rewritten as

$$\sum_{i=1}^{M} a_{mi}\Delta \boldsymbol{U}_i \Delta s_i = b_m \quad m=1,2,\cdots,M \tag{21}$$

where

$$\begin{cases} a_{mi} = \sum_{n=1}^{N} h^3 D(\boldsymbol{X}_m - \boldsymbol{x}_n) D(\boldsymbol{x}_n - \boldsymbol{X}_i) \\ b_m = \boldsymbol{U}_m^{\mathrm{B}} - \sum_{n=1}^{N} \boldsymbol{u}_n^* h^3 D(\boldsymbol{X}_m - \boldsymbol{x}_n) \end{cases} \tag{22}$$

Eq.(21) can be further reduced to

$$\sum_{i\in\{a_{mi}\neq 0\}} a_{mi}\Delta \boldsymbol{U}_i \Delta s_i = b_m \tag{23}$$

A crucial observation is that only when the nth Eulerian cell is influenced by both the mth and ith Lagrangian points (i.e., $\|\boldsymbol{X}_m - \boldsymbol{x}_n\| \leqslant R$ and $\|\boldsymbol{x}_n - \boldsymbol{X}_i\| \leqslant R$, where $R = O(h)$ is the influence radius of the interpolation function), $D(\boldsymbol{X}_m - \boldsymbol{x}_n) D(\boldsymbol{x}_n - \boldsymbol{X}_i) \neq 0$. Thus, to ensure $a_{mi}\neq 0$, we have

$$\|\boldsymbol{X}_m - \boldsymbol{X}_i\| \leqslant \|\boldsymbol{X}_m - \boldsymbol{x}_n\| + \|\boldsymbol{x}_n - \boldsymbol{X}_i\| \leqslant 2R = O(h) \tag{24}$$

When the distance between the mth and ith Lagrangian points satisfies the above restriction, applying a Taylor series expansion and conducting an error analysis yield

$$\Delta \boldsymbol{U}_i \Delta s_i = \Delta \boldsymbol{U}_m \Delta s_m + O(h^2) \tag{25}$$

Thus, Eq.(23) can be approximated as

$$\sum_{i\in\{a_{mi}\neq 0\}} a_{mi}\Delta \boldsymbol{U}_m \Delta s_m = b_m \tag{26}$$

which implies that the velocity correction at the mth Lagrangian point can be efficiently computed by

$$\Delta \boldsymbol{U}_m \Delta s_m = \frac{b_m}{\sum_{i\in\{a_{mi}\neq 0\}} a_{mi}} \tag{27}$$

This explicit approach eliminates computationally expensive matrix operations when evaluating velocity corrections, significantly improving computational efficiency, particularly in moving boundary problems. Zhao et al.[36] compared computational time consumption for different Lagrangian point counts in a simulation of flow past a transversely oscillating circular cylinder. Table 1 summarizes the CPU time per time step and IBM's contribution to total computational cost. Clearly, the computational cost of the explicit method[36] grows nearly linearly with the number of Lagrangian points, making it significantly more efficient than the original boundary condition-enforced IBM[32].

Table 1 Comparison of CPU time for different IBMs per time step with varying numbers of Lagrangian points in the simulation of flow past a transversely oscillating circular cylinder[36]

Number of Lagrangian points		100	250	500	1 000
Original boundary condition-enforced IBM	CPU time/ms	6.98	42.12	173.21	745.56
	Percentage/%	6.44	29.28	63.01	87.98
Multi-direct forcing IBM	CPU time/ms	1.15	5.16	20.64	52.49
	Percentage/%	1.12	4.83	16.87	34.04
Explicit boundary condition-enforced IBM	CPU time/ms	0.14	0.35	0.74	1.37
	Percentage/%	0.13	0.34	0.72	1.33

2.3 Local virtual body-fitted grid-based diffuse-interface IBM

The use of a smooth Dirac delta function for transferring boundary effects imposes certain limitations. Specifically, the object must be immersed in a uniform Eulerian mesh region, and a large number of uniformly distributed Lagrangian points are required due to the limited influence range of the Dirac delta function. To overcome these constraints, Du et al.[40] developed a local virtual body-fitted grid-based IBM, which enforced the no-slip boundary condition indirectly through a local virtual body-fitted grid. Based on the distribution of Lagrangian points, a local virtual body-fitted grid can be easily generated along the immersed boundary. As illustrated in Fig.4, this grid consists of two layers of virtual cells located inside and outside the immersed body. The Eulerian cells (represented by blue squares in Fig.4) covered by the virtual grid are responsible for receiving boundary influence from the virtual grid. In this method, the velocity correction is performed on the virtual grid. Before applying this correction, the predicted velocity field must first be interpolated from the underlying Eulerian mesh onto the virtual grid. Once the velocity field is corrected on the virtual grid, it is then mapped back to the Eulerian mesh to update the overall flow field.

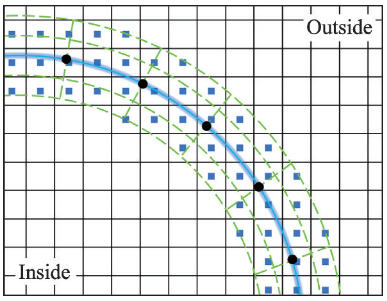


Fig.4 Schematic diagram of the virtual body-fitted grid for 2D case

For 2D case, the interaction between the underlying Eulerian mesh and the virtual grid can be handled by using bilinear interpolation, as the virtual grid is quadrilateral. However, in 3D case, the interpolation procedure requires modifications due to the irregular nature of the virtual grid. Specifically, the predicted velocity field can still be interpolated using a simple trilinear interpolation method, given that the Eulerian mesh remains Cartesian. However, to map the corrected velocity field from the virtual grid back to the Eulerian mesh, the IDW interpolation[38-39] is employed to accommodate the irregular spatial distribution.

The velocity correction procedure is conducted on the virtual body-fitted grid to enforce the no-slip boundary condition. Due to the irregularity of the virtual grid, the commonly used smooth Dirac delta function becomes unsuitable, necessitating the adoption of the IDW method[39]. For the mth Lagrangian point, its fluid velocity can be interpolated from the surrounding virtual grid points, given by

$$\begin{cases} \boldsymbol{U}_m^{l+1}=\sum\limits_{k=1}^{K}\omega_{mk}\boldsymbol{u}_{G,k}^{l+1} \\ \omega_{mk}=\dfrac{\gamma_{mk}}{\sum\limits_{k=1}^{K}\gamma_{mk}},\gamma_{mk}=\left(\dfrac{R-d_{mk}}{Rd_{mk}}\right)^2,R=\max(d_{mk}) \\ \qquad\qquad k=1,2,\cdots,K \end{cases} \tag{28}$$

where the subscript “G” represents the flow variable on the virtual grid; K denotes the number of virtual grid points associated with the mth Lagrangian point; and d_{mk} the distance between the mth Lagrangian point and its kth associated virtual grid point, i.e.

$$d_{mk}=\sqrt{(X_m-x_k)^2+(Y_m-y_k)^2+(Z_m-z_k)^2} \tag{29}$$

To enforce the no-slip boundary condition, the interpolated fluid velocity at the mth Lagrangian point must be equal to the boundary velocity, i.e.

$$\begin{cases} \boldsymbol{U}_m^{\mathrm{B}}=\boldsymbol{U}_m^{l+1}=\sum\limits_{n}\omega_{mn}\boldsymbol{u}_{G,n}^{l+1}=\sum\limits_{k}\omega_{mk}\boldsymbol{u}_{G,k}^{l+1} \\ \omega_{mn}=\begin{cases}\omega_{mk} & d_{mn}\leqslant R \\ 0 & d_{mn}>R\end{cases} \end{cases} \tag{30}$$

Conversely, the velocity correction at the nth virtual grid point is distributed from the velocity corrections at Lagrangian points as

$$\Delta\boldsymbol{u}_{G,n}=\sum_{m}\omega_{mn}\Delta\boldsymbol{U}_m \tag{31}$$

Then, the velocity at the nth virtual grid point is corrected by

$$\boldsymbol{u}_{G,n}^{l+1}=\boldsymbol{u}_{G,n}^{*}+\Delta\boldsymbol{u}_{G,n} \tag{32}$$

Substituting Eqs.(31—32) into Eq.(30), the following formula can be obtained as

$$\boldsymbol{U}_m^{\mathrm{B}}=\sum_{n}\omega_{mn}u_{G,n}^{*}+\sum_{n}\omega_{mn}\sum_{m}\omega_{mn}\Delta\boldsymbol{U}_m \tag{33}$$

which can be rewritten as

$$\boldsymbol{A}_G\boldsymbol{X}_G=\boldsymbol{B}_G \tag{34}$$

where

$$\boldsymbol{X}_G=[\Delta\boldsymbol{U}_1\quad\Delta\boldsymbol{U}_2\quad\cdots\quad\Delta\boldsymbol{U}_M]^{\mathrm{T}}$$

$$\boldsymbol{A}_G=\begin{bmatrix}\omega_{11} & \omega_{12} & \cdots & \omega_{1N} \\ \omega_{21} & \omega_{22} & \cdots & \omega_{2N} \\ \vdots & \vdots & & \vdots \\ \omega_{M1} & \omega_{M2} & \cdots & \omega_{MN}\end{bmatrix}\begin{bmatrix}\omega_{11} & \omega_{21} & \cdots & \omega_{M1} \\ \omega_{12} & \omega_{22} & \cdots & \omega_{M2} \\ \vdots & \vdots & & \vdots \\ \omega_{1N} & \omega_{2N} & \cdots & \omega_{MN}\end{bmatrix}$$

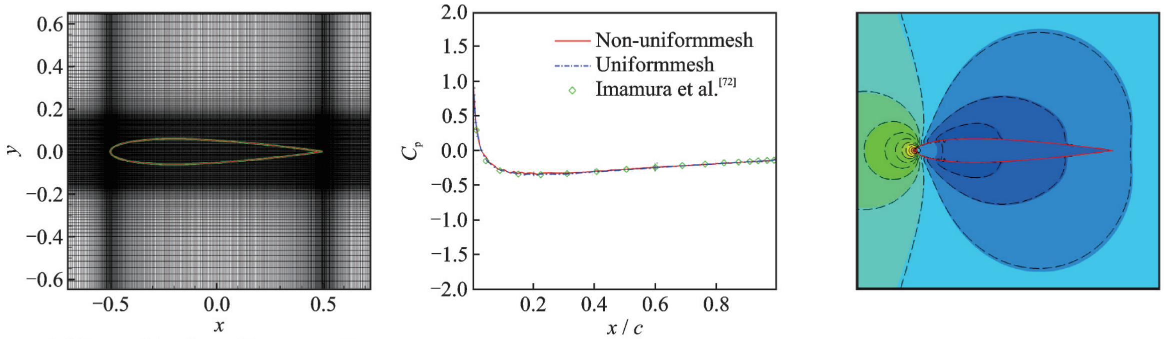
$$B_G=\begin{bmatrix}U_1^{\mathrm{B}}\\U_2^{\mathrm{B}}\\\vdots\\U_M^{\mathrm{B}}\end{bmatrix}-\begin{bmatrix}\omega_{11}&\omega_{12}&\cdots&\omega_{1N}\\\omega_{21}&\omega_{22}&\cdots&\omega_{2N}\\\vdots&\vdots&&\vdots\\\omega_{M1}&\omega_{M2}&\cdots&\omega_{MN}\end{bmatrix}\begin{bmatrix}\boldsymbol{u}_{G,1}^{*}\\\boldsymbol{u}_{G,2}^{*}\\\vdots\\\boldsymbol{u}_{G,N}^{*}\end{bmatrix}$$

By solving the above equation system (Eq.(34)) and using Eqs.(31,32), the velocity field on the virtual grid can be corrected. The corrected velocity field is then interpolated back onto the underlying Eulerian mesh to update the flow field.

The local virtual grid, to some extent, decouples the spacing of Lagrangian points from that of the Eulerian mesh, enabling a more flexible non-uniform distribution of Lagrangian points. Compared to the original boundary condition-enforced IBM[32-33], this method can accurately enforce the no-slip boundary condition with fewer Lagrangian points. Furthermore, it allows objects to be immersed in a non-uniform Eulerian mesh region, which is particularly useful for practical engineering applications. Du et al.[40] tested this method by simulating flow around a NACA0012 airfoil at $Re=500$ using two sets of meshes, as summarized in Table 2. The term "Uniform mesh" refers to the airfoil discretized using uniformly distributed Lagrangian points and immersed in a uniform Eulerian mesh region. In contrast, "Non-uniform mesh" indicates that the airfoil has non-uniformly distributed Lagrangian points and is immersed in a non-uniform Eulerian mesh region. As shown in Table 1, the non-uniform mesh uses fewer Lagrangian points, leading to reduced computational cost. Fig.5 illustrates a NACA0012 airfoil immersed in a non-uniform Eulerian mesh region, with the obtained pressure distribution matching that of the uniform mesh case[40].

Table 2 Comparison of different meshes for the flow around a NACA0012 airfoil at *Re*=500[40]

Mesh	Uniform mesh	Non-uniform mesh
Mesh size	681×301	501×301
Minimum Mesh spacing *h*	0.002 5	0.002 5 (near the leading and trailing edges)
Minimum Lagrangian mesh spacing	0.006 36	0.006 25 (near the leading and trailing edges)
Number of Lagrangian points	320	160



(a) Zoomed-in view of the non-uniform Eulerian meshes

(b) Comparison of the pressure coefficient C_p distribution along the surface of the airfoil

(c) Pressure contours around the airfoil (Contour: results of uniform mesh; Black dotted lines: results of non-uniform mesh)

Fig.5 Non-uniform mesh and pressure distribution for the flow around a NACA0012 airfoil at $Re=500$[40]

2.4 Diffuse-interface IBM for incompressible thermal flows

Currently, the diffuse-interface IBM has been extensively applied for simulating incompressible thermal flows, incorporating temperature boundary conditions such as isothermal and iso-heat-flux conditions. The conventional diffuse-interface IBM can easily deal with the isothermal boundary condition since it is a Dirichlet-type condition, similar to the no-slip boundary condition. For instance, Zhang et al.[42] and Ren et al.[43] extended their IBM frameworks to simulate thermal flows by introducing a heat source term in the energy equation and implementing the isothermal boundary condition in the same manner as the no-slip condition. In contrast, the iso-heat-flux boundary condition, classified as a Neumann-type condition, is more challenging to implement, which is the focus of this section. Ren et al.[44] proposed a heat flux correction-based IBM that directly adjusted the temperature field by using the offset of the given normal temperature derivative at the boundary and the predicted value. However, the Neumann boundary condition is only approximately satisfied by their heat flux correction procedure[73]. Due to its simplicity and efficiency, Suzuki et al.[74] further extended this heat flux correction method[44] to simulate thermal flows with moving boundaries. Different from Ren et al.[44], Hu et al.[75] derived the jump conditions for the temperature derivative to enforce the Neumann boundary condition. In their method, the difference between the normal derivatives on either side of the interface contributes as a heat flux, which is then distributed to surrounding Eulerian points to correct the temperature field. In addition, Guo et al.[76] proposed an alternative heat flux correction-based IBM, in which the Eulerian point is defined at the center of the cell face. This method enforces the heat flux boundary condition by correcting the temperature gradient at the cell face center.

Some other approaches utilize auxiliary layers to enforce the Neumann boundary condition. For instance, Zhang et al.[42] introduced an auxiliary layer outside the boundary and approximated the boundary temperature from the heat flux using a finite difference scheme. Furthermore, Wang et al.[45] extended this idea by incorporating two auxiliary layers to approximate the iso-heat-flux condition. They also employed a fractional-step technique to solve the governing equations (Eq.(2)). For thermal flows, both the velocity and temperature fields must be corrected to account for the immersed boundary's thermal effects. The temperature correction is obtained by solving

$$\frac{\partial T}{\partial t}=q \tag{35}$$

which can be discretized as

$$T^{l+1}=T^{*}+\Delta tq=T^{*}+\Delta T \tag{36}$$

To enforce the iso-heat-flux condition, two auxiliary layers of Lagrangian points are positioned on either side of the solid boundary, as illustrated in Fig.6[45]. The constant heat flux at the boundary is approximated by

$$\frac{\partial T_m^{\mathrm{B}}}{\partial n}=\frac{1}{2h}\left(T_m^{\mathrm{O},l+1}-T_m^{\mathrm{I},l+1}\right) \tag{37}$$

where the superscripts "O" and "I" represent the outer and inner auxiliary layers, respectively. The temperature of the mth Lagrangian point in the auxiliary layer is interpolated from the temperature of the surrounding Eulerian cells, i.e.

$$T_m^{\mathrm{O},l+1}=\sum_n T_n^{l+1}h^3 D\left(\boldsymbol{X}_m^{\mathrm{O}}-\boldsymbol{x}_n\right) \tag{38a}$$

$$T_m^{\mathrm{I},l+1}=\sum_n T_n^{l+1}h^3 D\left(\boldsymbol{X}_m^{\mathrm{I}}-\boldsymbol{x}_n\right) \tag{38b}$$

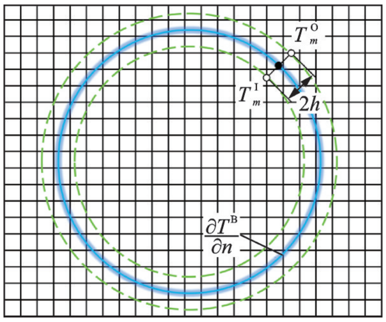


Fig.6 Illustration of auxiliary layers for iso-heat-flux boundary condition[45]

According to the work of Wang et al.[45], only the temperature correction on the inner layer is selected and subsequently distributed to the Eulerian cells inside the inner layer. The corresponding expression is given by

$$\Delta T_n=\begin{cases}\sum_m \Delta T_m^{\mathrm{I}}\Delta s_m^{\mathrm{I}} D\left(\boldsymbol{x}_n-\boldsymbol{X}_m^{\mathrm{I}}\right) & \boldsymbol{x}_n\in\Omega_{\mathrm{I}}\\ 0 & \boldsymbol{x}_n\notin\Omega_{\mathrm{I}}\end{cases} \tag{39}$$

where Ω_{I} represents the domain inside the inner layer. The combination of Eqs.(36—39) yields

$$2h\frac{\partial T_m^{\mathrm{B}}}{\partial n}-T_m^{\mathrm{O},*}+T_m^{\mathrm{I},*}=\sum_n h^3\left(D\left(\boldsymbol{X}_m^{\mathrm{O}}-\boldsymbol{x}_n\right)-D\left(\boldsymbol{X}_m^{\mathrm{I}}-\boldsymbol{x}_n\right)\right)\sum_m \Delta T_m^{\mathrm{I}}\Delta s_m^{\mathrm{I}} D\left(\boldsymbol{x}_n-\boldsymbol{X}_m^{\mathrm{I}}\right) \tag{40}$$

where

$$T_m^{\mathrm{O},*}=\sum_n T_n^{*}h^3 D\left(\boldsymbol{X}_m^{\mathrm{O}}-\boldsymbol{x}_n\right) \tag{41a}$$

$$T_m^{\mathrm{I},*}=\sum_n T_n^{*}h^3 D\left(\boldsymbol{X}_m^{\mathrm{I}}-\boldsymbol{x}_n\right) \tag{41b}$$

Eq.(40) can be rewritten in a matrix form as

$$\boldsymbol{CY}=\boldsymbol{D} \tag{42}$$

where

$$\boldsymbol{Y}=\left[\Delta T_1^{\mathrm{I}}\Delta s_1^{\mathrm{I}}\quad \Delta T_2^{\mathrm{I}}\Delta s_2^{\mathrm{I}}\quad \cdots \quad \Delta T_M^{\mathrm{I}}\Delta s_M^{\mathrm{I}}\right]^{\mathrm{T}}$$

$$\boldsymbol{C}=h^3\begin{bmatrix} D_{11}^{\mathrm{O}}-D_{11}^{\mathrm{I}} & D_{12}^{\mathrm{O}}-D_{12}^{\mathrm{I}} & \cdots & D_{1N}^{\mathrm{O}}-D_{1N}^{\mathrm{I}} \\ D_{21}^{\mathrm{O}}-D_{21}^{\mathrm{I}} & D_{22}^{\mathrm{O}}-D_{22}^{\mathrm{I}} & \cdots & D_{2N}^{\mathrm{O}}-D_{2N}^{\mathrm{I}} \\ \vdots & \vdots & & \vdots \\ D_{M1}^{\mathrm{O}}-D_{M1}^{\mathrm{I}} & D_{M2}^{\mathrm{O}}-D_{M2}^{\mathrm{I}} & \cdots & D_{MN}^{\mathrm{O}}-D_{MN}^{\mathrm{I}} \end{bmatrix}\cdot \begin{bmatrix} D_{11}^{\mathrm{I}} & D_{12}^{\mathrm{I}} & \cdots & D_{1M}^{\mathrm{I}} \\ D_{21}^{\mathrm{I}} & D_{22}^{\mathrm{I}} & \cdots & D_{2M}^{\mathrm{I}} \\ \vdots & \vdots & & \vdots \\ D_{N1}^{\mathrm{I}} & D_{N2}^{\mathrm{I}} & \cdots & D_{NM}^{\mathrm{I}} \end{bmatrix}$$

$$\boldsymbol{D}=2h\begin{bmatrix} \dfrac{\partial T_1^{\mathrm{B}}}{\partial n} \\ \dfrac{\partial T_2^{\mathrm{B}}}{\partial n} \\ \vdots \\ \dfrac{\partial T_M^{\mathrm{B}}}{\partial n} \end{bmatrix}-h^3\begin{bmatrix} D_{11}^{\mathrm{O}} & D_{12}^{\mathrm{O}} & \cdots & D_{1N}^{\mathrm{O}} \\ D_{21}^{\mathrm{O}} & D_{22}^{\mathrm{O}} & \cdots & D_{2N}^{\mathrm{O}} \\ \vdots & \vdots & & \vdots \\ D_{M1}^{\mathrm{O}} & D_{M2}^{\mathrm{O}} & \cdots & D_{MN}^{\mathrm{O}} \end{bmatrix}\begin{bmatrix} T_1^* \\ T_1^* \\ \vdots \\ T_N^* \end{bmatrix}+ h^3\begin{bmatrix} \mathrm{D}_{11}^{\mathrm{I}} & D_{12}^{\mathrm{I}} & \cdots & D_{1N}^{\mathrm{I}} \\ D_{21}^{\mathrm{I}} & D_{22}^{\mathrm{I}} & \cdots & D_{2N}^{\mathrm{I}} \\ \vdots & \vdots & & \vdots \\ D_{M1}^{\mathrm{I}} & D_{M2}^{\mathrm{I}} & \cdots & D_{MN}^{\mathrm{I}} \end{bmatrix}\begin{bmatrix} T_1^* \\ T_1^* \\ \vdots \\ T_N^* \end{bmatrix}$$

By solving the above equation system (Eq.(42)) and utilizing Eq.(39), the temperature corrections for the Eulerian cells inside the inner layer are determined, allowing the temperature field to be updated using Eq.(36). This method effectively simulates thermal flows with the Neumann boundary condition.

An important characteristic of the above method[45] is its biased distribution of the heat source term. Du et al.[46] incorporated this concept into their virtual body-fitted grid-based IBM for simulating thermal flows with the Neumann boundary condition. Utilizing the local virtual body-fitted grid, they adopted a simpler and more direct approach using quadratic function fitting. As shown in Fig.7[46], layers 1 and 2 lie inside the immersed body, while layer 3 represents the immersed boundary. The temperatures in these layers are treated as unknowns, whereas the temperatures in layers 4 and 5, located outside the immersed body, are considered known. The normal spacing between these virtual layers is $\Delta\eta$. Assuming a quadratic temperature distribution along the normal direction of the wall

$$T(\eta)=a\eta^2+b\eta+c \tag{43}$$

where η represents the normal position relative to the boundary. Given the heat flux at the boundary $\dfrac{\partial T^{\mathrm{B}}}{\partial n}$, we obtain

$$\left.\frac{\mathrm{d}T}{\mathrm{d}\eta}\right|_{\eta=0}=b=\frac{\partial T^{\mathrm{B}}}{\partial n} \tag{44}$$

Substituting the temperatures of layers 4 and 5 into Eq.(43) results in

$$T_4=a(\Delta\eta)^2+b\Delta\eta+c \tag{45a}$$

$$T_5=4a(\Delta\eta)^2+2b\Delta\eta+c \tag{45b}$$

Solving Eqs.(44, 45) yields

$$\begin{cases} a=\dfrac{1}{3(\Delta\eta)^2}\left(T_5-T_4-\Delta\eta\dfrac{\partial T^{\mathrm{B}}}{\partial n}\right) \\ b=\dfrac{\partial T^{\mathrm{B}}}{\partial n} \\ c=\dfrac{1}{3}\left(4T_4-T_5-2\Delta\eta\dfrac{\partial T^{\mathrm{B}}}{\partial n}\right) \end{cases} \tag{46}$$

Consequently, the temperatures at layers 1, 2, and 3 can be calculated by

$$T_1=4a(\Delta\eta)^2-2b\Delta\eta+c \tag{47a}$$

$$T_2=a(\Delta\eta)^2-b\Delta\eta+c \tag{47b}$$

$$T_3=c \tag{47c}$$

Finally, the temperatures of the Eulerian cells (blue squares in Fig.7) covered by the virtual grid inside the boundary are corrected using the temperatures at layers 1, 2, and 3.

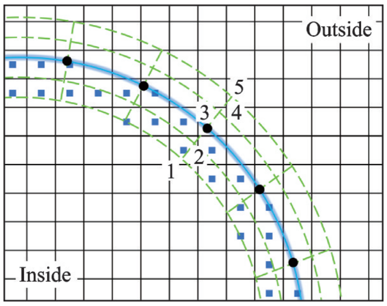


Fig.7 Illustration of the virtual body-fitted grid for iso-heat-flux boundary condition[46]

The forced convection over a stationary cylinder is a benchmark case widely studied in the literature. In this simulation, the circular cylinder transfers heat outward under a non-dimensional heat flux condition $\dfrac{\partial T}{\partial n}=-1$. Wang et al.[45], Suzuki et al.[74], Guo et al.[76], and Du et al.[46] all simulated this case using their respective thermal IBMs. Fig.8 presents the distribution of the local Nusselt number Nu along the cylinder surface[46]. The results indicate a gradual decrease in the local Nusselt number from the leading edge (0°) to the trailing edge (180°). The agreement between results obtained from different IBMs[45-46, 74] demonstrates consistency and reliability. Fig.9 displays isothermal contours for different Reynolds numbers, including the results from Guo et al.[76] and Du et al.[46]. As the Reynolds number increases, enhanced convection lowers the temperature around the cylinder, further validating the accuracy of these methods.

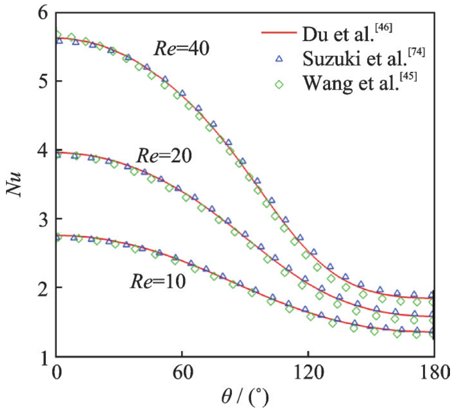
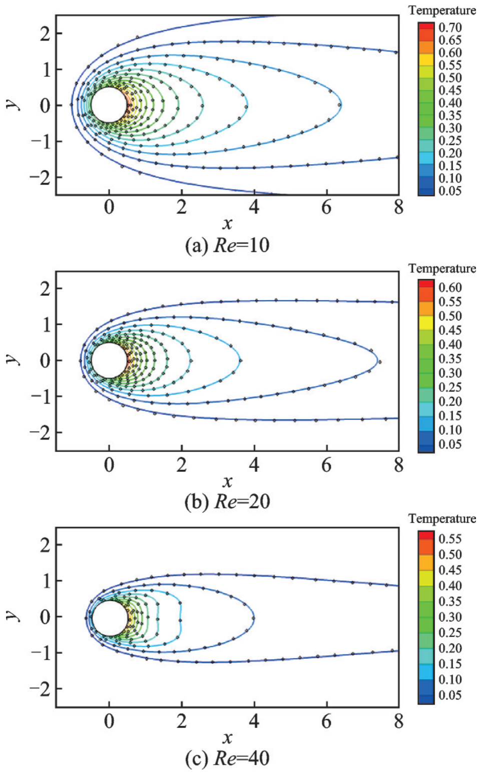


Fig.8 Local Nusselt number along the cylinder surface for the forced convection over an iso-heat-flux circular cylinder[46]



(a) Re=10

(b) Re=20

(c) Re=40

Isotherms: results of Du et al.[46]

Black diamond: results of Guo et al.[76]

Fig.9 Comparison of isotherms for the forced convection over an iso-heat-flux circular cylinder at Re = 10, 20, and 40

2.5 Diffuse-interface IBM for incompressible turbulent flows

Simulating turbulent flows at high Reynolds numbers has long been a challenge for the diffuse-interface IBM. These flows exhibit thin boundary layers with steep velocity gradients, necessitating high mesh resolution. However, in IBM, refining the Cartesian mesh exclusively in the normal direction of a curved wall is not feasible. To mitigate this issue, wall models[77-80] have been widely adopted, reducing the need for excessive mesh refinement. In the literature, numerous approaches have integrated wall models with the sharp-interface IBM[50-55] for high-Reynolds-number turbulence simulations, leveraging its straightforward velocity reconstruction at interface nodes. However, this comes at the cost of a complex and labor-intensive mesh identification process. In contrast, the diffuse-interface IBM eliminates the need for mesh cell identification but introduces additional complexity due to the smooth interpolation function governing flow field interactions with the boundary, complicating the integration of wall models. To address this, recent methods[57, 61] based on boundary condition-enforced IBM have been developed.

Shi et al.[56] linked the tangential force term to the wall shear stress by integrating the momentum equation in the wall-normal direction. Building on this, Du et al.[57] incorporated this approach into boundary condition-enforced IBM[32-33], simplifying the tangential force term to

$$F_\tau \approx \frac{\tau_\mathrm{w}}{\Delta_P} - \frac{\tau_P}{\Delta_P} \tag{48}$$

where τ_w is the wall shear stress and τ_P the shear stress at the point P. Unlike Shi et al.[56], who retained only the first term on the right-hand side of Eq.(48), Du et al.[57] refined the formulation. Here, the subscript "τ" represents the tangential direction of the wall. The integral length Δ_P and the shear stress τ_P are determined using Reichardt's law and Spalding's formula[81].

To incorporate wall models, Du et al.[57] introduced two auxiliary layers outside the wall, consisting of a series of Lagrangian points: The reference layer and the enforced layer. As shown in Fig.10, the wall model is used at the reference layer (point F) to calculate the wall shear stress τ_{w}, while the boundary condition is enforced at the enforced layer (point Q)[57]. The vector $\boldsymbol{B}$ in Eq.(20) can be decomposed in the local orthogonal coordinate system as

$$\boldsymbol{B} = \boldsymbol{B}_\tau \boldsymbol{e}_\tau + \boldsymbol{B}_n \boldsymbol{e}_n \tag{49}$$

where

$$\boldsymbol{B}_\tau = \Delta t \boldsymbol{F}_\tau = \left[\Delta t\left(\frac{\tau_{\mathrm{w}}}{\Delta_P} - \frac{\tau_P}{\Delta_P}\right)\right]_{M\times 1} \tag{50a}$$

$$\boldsymbol{B}_n = \Delta t \boldsymbol{F}_n = \left[\bar{V}^{\mathrm{B}} - \bar{V}^{*}\right]_{M\times 1} \tag{50b}$$

where the subscript "n" represents the normal direction of the wall, and "$\bar{V}$" the normal velocity component. The unit vectors $\boldsymbol{e}_\tau$ and $\boldsymbol{e}_n$ correspond to the tangential and normal directions, respectively. The normal velocity at the enforced layer is reconstructed using a parabolic distribution

$$\bar{V}_Q^{\mathrm{B}} = \frac{\Delta_Q^2}{\Delta_F^2}\bar{V}_F \tag{51}$$

where Δ_F and Δ_Q are the distances from points F and Q to the wall, respectively. The components $\boldsymbol{B}_\tau$ and $\boldsymbol{B}_{\mathrm{n}}$ are then transformed back into vector $\boldsymbol{B}$ in Eq.(20), ensuring that the wall shear stress is enforced within the boundary condition-enforced IBM framework[32-33]. Du et al.[57] used this wall model-based diffuse-interface IBM to perform the RANS simulations of high Reynolds number turbulent flows, including the flow around NACA23012 airfoil at $Re = 1.88\times10^6$. Fig.11 shows the computed pressure coefficient C_{p} and skin friction coefficient C_{f} distributions, compared against reference results from a body-fitted method[57]. The developed IBM demonstrates good agreement with the reference data, except near the leading edge, where slight discrepancies appear due to the thinner boundary layer. This thinning may cause the auxiliary layer to extend beyond the boundary layer, leading to deviations.

In addition, Shi et al.[82] and Yan et al.[61] incorporated a non-equilibrium wall model, utilizing the velocity at the reference layer to establish a tangential slip velocity at the Lagrangian point

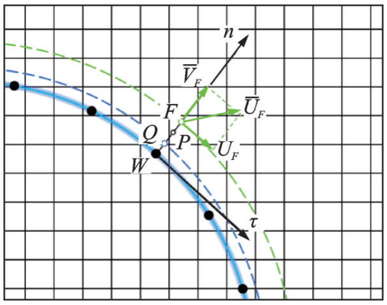
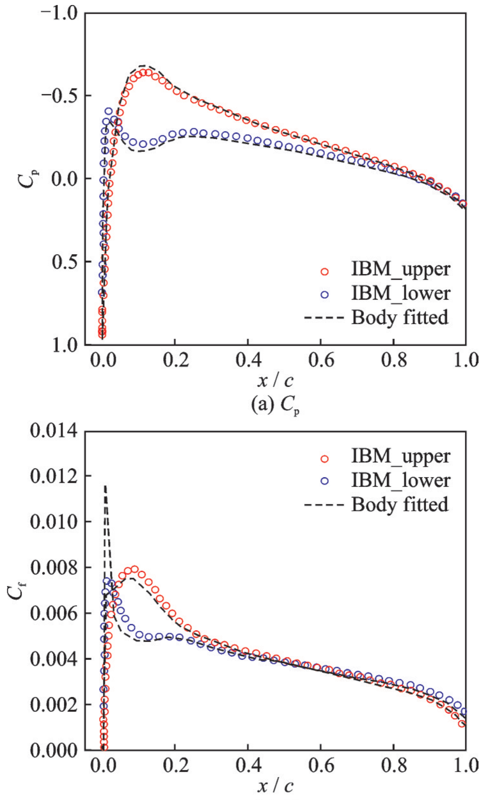


Fig.10 Illustration of the local orthogonal coordinate system and auxiliary layers[57]



(a) C_{p}

(b) C_{f}

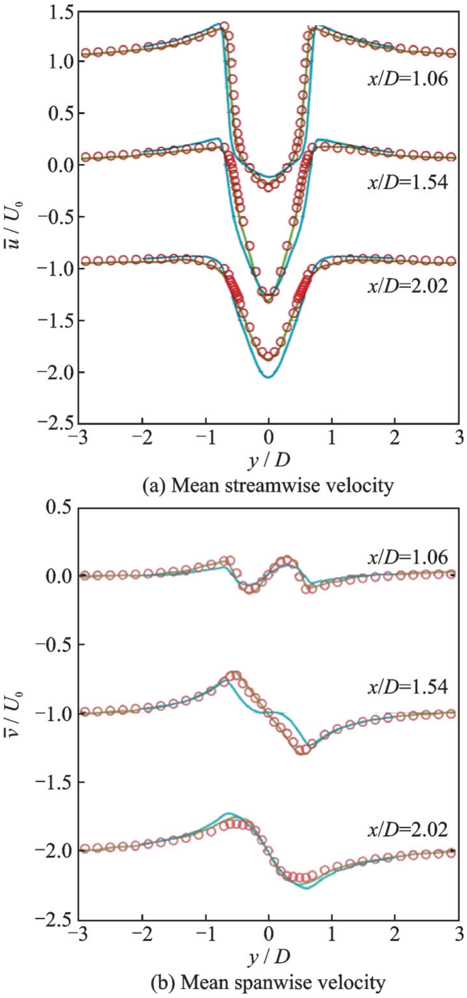
Fig.11 Distributions of pressure coefficient C_{p} and skin friction coefficient C_{f} along the surface of the NACA23012 airfoil at $Re = 1.88\times10^6$ [57]

$$\bar{U}_W^{\mathrm{B}} = \bar{U}_F - \Delta_F \left.\frac{\partial \bar{U}}{\partial n}\right|_{\Delta_F} \tag{52}$$

where "$\bar{U}$" denotes the tangential velocity component. The normal gradient of tangential velocity $\left.\frac{\partial \bar{U}}{\partial n}\right|_{\Delta_F}$ is determined by integrating the turbulent boundary layer (TBL) equation

$$\left.\frac{\partial \bar{U}}{\partial n}\right|_{\Delta_F} = \frac{\tau_{\mathrm{w}} + S\Delta_F}{\rho(\nu + \nu_t)} \tag{53}$$

where the term S is simplified to the pressure gradient term only. Yan et al.[61] enforced the tangential slip velocity and the normal non-penetration condition at the boundary using an explicit boundary condition-enforced IBM[36]. Leveraging the LES, they used their wall-modeling diffuse-interface IBM to simulate turbulent flow past a circular cylinder at $Re = 3\,900$. Fig.12 compares the mean velocity profiles at three locations in the cylinder wake[61,83-84], where U_0 is the free-stream velocity. The results show a transformation from a U-shaped to a V-shaped mean streamwise velocity profile, while the mean spanwise velocity exhibits an asymmetric pattern. These observations are consistent with previous studies[83-84]. Fig.13 depicts the 3D wake structures identified using the Q-criterion, capturing key flow features such as separation, free shear layer transition, vortex shedding, and the multiscale nature of wake dynamics[61].



(a) Mean streamwise velocity

(b) Mean spanwise velocity

Red circles: results of Yan et al.[61]

Solid green line: results of Kravchenko et al.[83]

Solid blue line: results of Parnaudeau et al.[84]

Fig.12 Comparison of the mean velocity profiles at three locations in the wake of a circular cylinder at $Re = 3\,900$

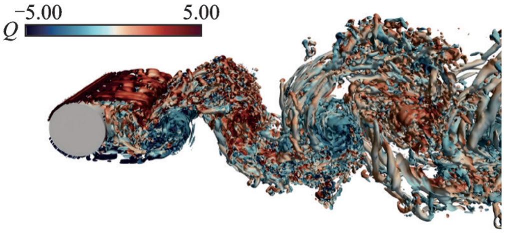


Fig.13 Instantaneous Q-isosurface of wake flows for the flow past a circular cylinder at $Re = 3\,900$[61]

It is important to note that an auxiliary layer outside the wall is crucial for computing wall shear stress using wall models. Various wall models are available for turbulent flow simulations, including wall functions (logarithmic and power laws[80]) and the wall stress model (also known as the two-layer model)[77, 79, 85]. Once an appropriate wall model is selected, determining the location of the reference layer becomes essential, as it directly impacts the accuracy of the computed wall shear stress. The reference layer must be positioned sufficiently close to the wall to capture boundary layer effects while remaining outside the boundary influence region to maintain numerical stability. Currently, its placement is typically determined artificially. For example, Ma et al.[58] and Yan et al.[61] both set $\Delta_F = 3h$ in their simulations.

3 Some Applications

This section presents several applications of the diffuse-interface IBM in simulating complex moving boundary problems, showcasing its exceptional performance.

3.1 Case 1: Flow around a flapping dragonfly

The first application primarily demonstrates the advantages of IBM in handling complex moving boundary problems. Simulating the flow around flapping insects presents significant challenges for conventional body-fitted mesh methods due to the intricate interactions between the wings and the body. The dynamic motion of the wings makes generating body-fitted meshes particularly difficult. However, the diffuse-interface IBM effectively addresses this issue by utilizing a fixed Cartesian mesh. Wu and

Shu[86] and Yang et al.[87] employed the boundary condition-enforced IBM to simulate the flow around a flapping dragonfly. They used a simplified dragonfly model consisting of a stationary body and two pairs of wings (forewings and hindwings), as illustrated in Fig.14[87]. In their simulation, the wings undergo sinusoidal pitching-rolling motions, and the Reynolds number is set to 500, based on the span length of the forewing.

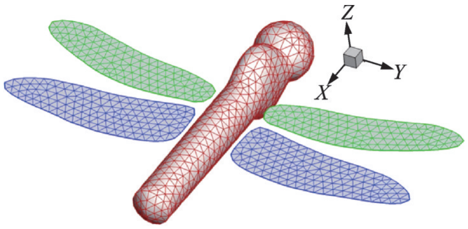
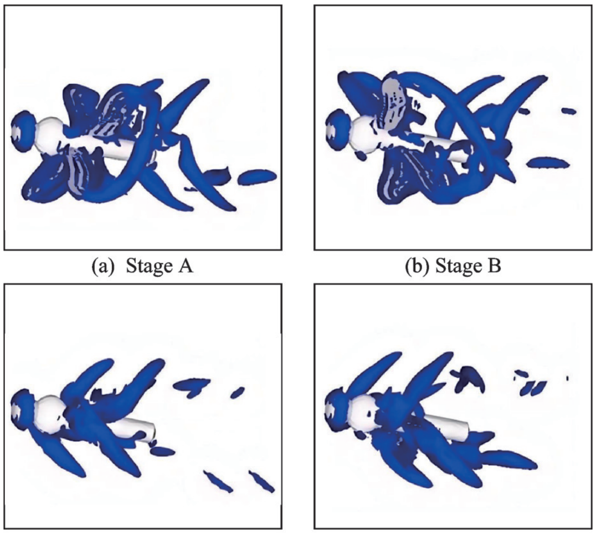


Fig.14 Illustration of a simplified dragonfly model[87]

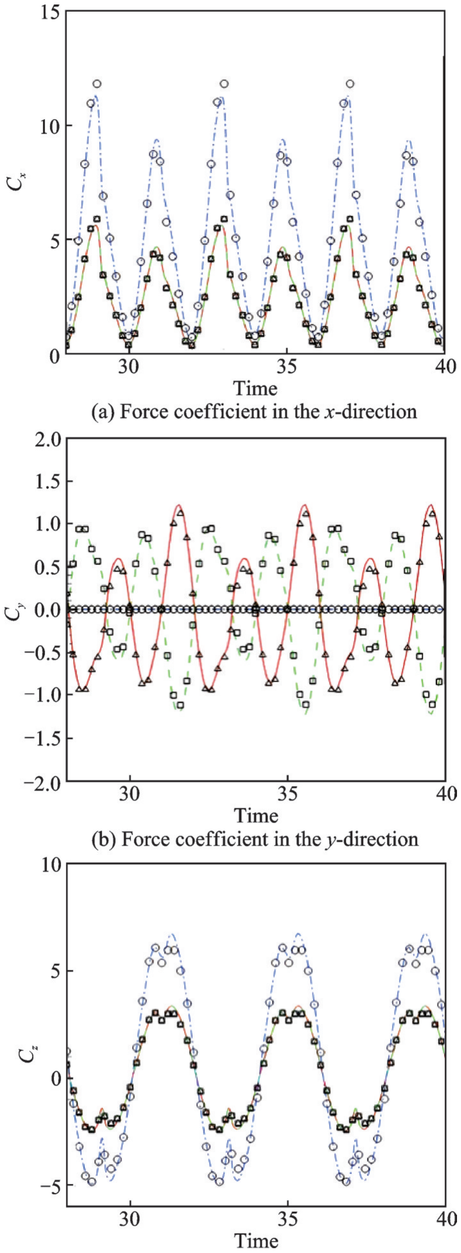
Fig.15 presents the 3D vortical structures surrounding the flapping dragonfly at different stages of motion[86] . At Stages A and B, the wings reach their extreme pitching positions and balanced rolling positions, generating dominant tip and wake vortices. At Stages C and D, the wings are at their balanced pitching positions and extreme rolling positions, where detached leading-edge vortices become visible on the wing surface. The strength of these vortices increases closer to the wingtip. The evolution of force coefficients in the x-, y-, and z-directions for the forewings is displayed in Fig.16. The figure plots the force coefficients for an individual forewing as well as the combined force coefficients for both wings. Due to the dragonfly's symmetry about the $y = 0$ plane, the force coefficients in the x- and z-directions are identical for both forewings, while those in the y-direction are opposite. This numerical example highlights the capability of the diffuse-interface IBM in accurately capturing the complex flow structures around flapping-wing in-



(a) Stage A (b) Stage B

(c) Stage C (d) Stage D

Fig.15 3D vortical structures for the flow around a flapping dragonfly at different stages[86]



(a) Force coefficient in the x-direction

(b) Force coefficient in the y-direction

(c) Force coefficient in the z-direction

Results from Wu and Shu[86]: triangles for forewing 1, squares for forewing 2, and circles for both forewings combined;

Results from Yang et al.[87]: solid red line for forewing 1, green dashed line for forewing 2, and blue dashed-dotted line for both forewings combined

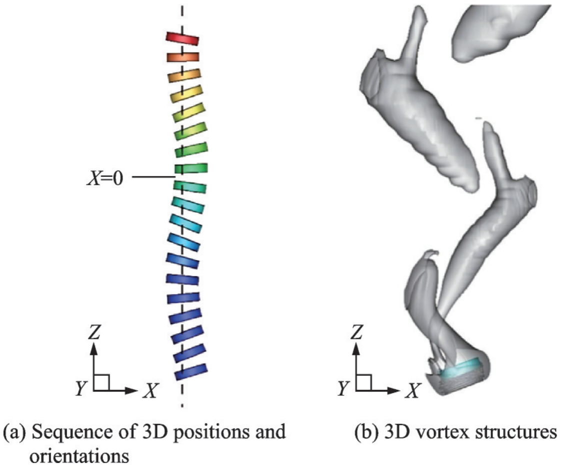
Fig.16 Force coefficients for the forewings in the flow around a flapping dragonfly

sects, demonstrating its potential for simulating bio-inspired aerodynamics.

3.2 Case 2: Freely falling disk

The second application highlights the advantages of IBM in addressing fluid-structure interaction (FSI) problems. The freely falling disk problem is a classic FSI case characterized by complex dynamic behavior. Wang et al.[88] incorporated the boundary condition-enforced IBM[33] into a moving Cartesian frame to effectively simulate the 3D motion of a freely falling disk in an unbounded domain. The moving Cartesian mesh is assigned the same translational velocity as the falling disk, allowing the method to handle freely moving objects in an infinite domain. Additionally, IBM eliminates the need for the tedious re-meshing process required in conventional arbitrary Lagrangian-Eulerian (ALE) approaches.

Wang et al.[88] studied the fluttering motion of a disk with an aspect ratio $Ar = 1/4$, initially released with a tilt angle of 0.2 rad, as illustrated in Fig.17(a). Here, Ar represents the ratio of the disk thickness to its diameter. The Reynolds number is set to 240, which is defined based on the disk diameter and its average terminal falling velocity. This motion is essentially 2D and can be quantitatively characterized by four properties: The Strouhal number (St), maximum lateral displacement (Δx), maximum inclination angle ($\theta_{\max}$), and the phase difference ($\Delta\psi$) between the x-velocity and the inclination angle. Fig.17(b) further illustrates the instantaneous vortex structures, revealing a series of symmetric hairpin vortices shed into the wake as the disk oscillates from side to side. Table 3 presents a comparison of these properties with experimental data[89] and previous numerical solutions[90], demonstrating the strong agreement between the results of Wang et al.[88] and experimental observations. Additionally, Wang et al.[88] examined different falling modes of a disk with $Ar = 0$, which was theoretically regarded as a flat cylinder with zero thickness. Two representative falling modes and their corresponding instantaneous vortex structures are shown in Fig.18[88], including tumbling and spiral motions. The left figures show sequences of 3D positions and orientations, while the right figures depict 3D vortex structures. These numerical experiments highlight the superior performance of the diffuse-interface IBM in handling complex FSI problems.

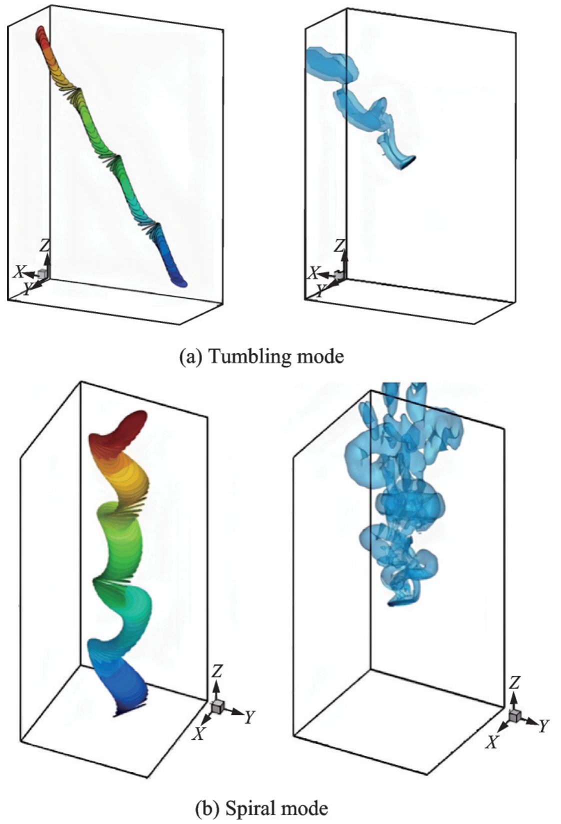


(a) Sequence of 3D positions and orientations
(b) 3D vortex structures

Fig.17 Motion of a freely falling disk with $Ar = 1/4$ at $Re = 240$[88]

Table 3 Comparison of four properties for a freely falling disk with $Ar = 1/4$

Properties	St	Δx	$\theta_{\max}/(°)$	$\Delta\psi/(°)$
Fernandes et al.[89] (Exp.)	0.122	0.15	22.72	195.3
Shenoy and Kleinstreuer[90]	0.171	0.159	27.51	—
Wang et al.[88]	0.133	0.154	24.69	191.8



(a) Tumbling mode
(b) Spiral mode

Fig.18 Two motion modes of a freely falling disk with $Ar = 0$[88]

3.3 Case 3: Self-propulsion of flapping wing

The final application demonstrates the use of IBM in more practical engineering problems. Flapping-wing-based propulsion is widely observed in nature, exemplified by the swimming of fish and the flight of birds. To gain deeper insights into the self-propulsion mechanism of flapping wings, Lin et al. [91] numerically investigated the hydrodynamic behavior of an unconstrained flapping foil using the boundary condition-enforced IBM[33]. Their simulations employed a simplified pitching foil model, as illustrated in Fig. 19[91]. Here, c and b denote the chord length and thickness of the foil, respectively. The foil undergoes a pitching motion $\theta(t) = \theta_m \sin(2\pi ft)$ with its pivot fixed at $x/c = 0.05$, where f and θ_m represent the pitching frequency and amplitude. The Reynolds number, defined based on the chord length, is set to 200.

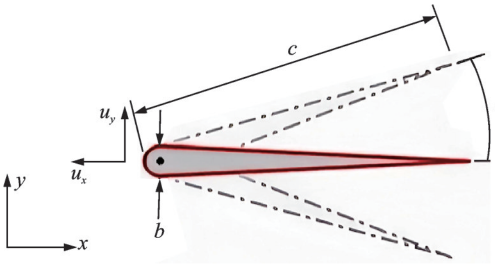
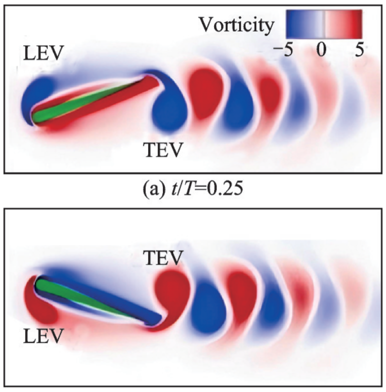


Fig.19 Schematic diagram of a simplified pitching foil model[91]

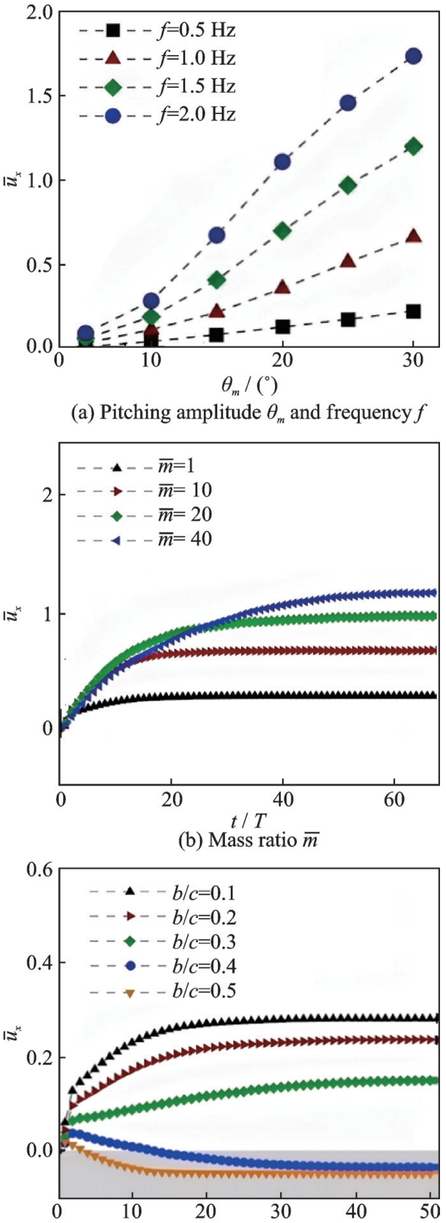
In their simulations, the unconstrained pitching foil achieves self-propulsion in the longitudinal direction while passively oscillating laterally. During this self-propelled motion, the pivot point moves downward as the foil pitches upward, and vice versa. Consequently, the leading-edge vortex (LEV) rotates in the same direction as the trailing-edge vortex (TEV), as clearly observed in Fig.20[91], which depicts the instantaneous vorticity contours. Lin et al.[91] analyzed the effects of various parameters on the self-propulsion of the pitching foil, including the pitching frequency f, pitching amplitude θ_m, mass ratio $\bar{m} = m/m_\mathrm{f}$ (where m_f is the fluid mass with the equivalent area of the foil), and thickness-to-chord ratio b/c. Fig.21 indicates that the mean longitudinal speed $\bar{u}_x$ of the pitching foil increases significantly with higher pitching ampli-



(a) t/T=0.25

(b) t/T=0.75

Fig.20 Instantaneous vorticity contours for the pitching foil[91]



(a) Pitching amplitude θ_m and frequency f

(b) Mass ratio $\bar{m}$

(c) Thickness-to-chord ratio b/c

Fig.21 Variation of the mean longitudinal speed $\bar{u}_x$ with different parameters[91]

tudes θ_m, higher pitching frequencies f, or larger mass ratios $\bar{m}$[91]. However, an increase in the thickness-to-chord ratio b/c results in a reduction in the longitudinal speed $\bar{u}_x$.

Similar to the freely falling disk problem, simulating the self-propulsion of a flapping wing poses significant challenges for conventional body-fitted methods due to the necessity of continuous re-meshing. By employing the diffuse-interface IBM, Lin et al.[91] successfully simulated the self-propelled flapping foil and investigated its hydrodynamic behavior, demonstrating the excellent capability of diffuse-interface IBM in handling complex moving boundary problems.

4 Conclusions and Outlook

The diffuse-interface IBM has gained significant popularity over the past decades due to its exceptional capability in simulating flows around complex geometries and moving boundaries, demonstrating its broad applicability across various fields, including aerospace, marine engineering, and biological flow studies. A key advantage of this method is its simplicity, making it easy to understand, implement, and seamlessly integrate into various flow solvers to address diverse flow problems. Over years of development, numerous variants have emerged, extending their applicability from isothermal to thermal flows and from laminar to turbulent flows. Moreover, the method effectively handles both Dirichlet and Neumann boundary conditions. This paper presents several variants of the diffuse-interface IBM and showcases selected applications, with a particular focus on the contributions of the authors' group.

Despite its advantages, a primary limitation of the diffuse-interface IBM arises in high Reynolds number flows, where a thin boundary layer forms. Since Cartesian meshes cannot be refined in the wall-normal direction for curved immersed boundaries, accurately resolving the near-wall region remains a challenge. The incorporation of wall models offers a promising strategy to mitigate this issue. However, the smooth interpolation function used to couple the flow field with the boundary complicates the implementation of these models. Research in this area remains relatively limited, highlighting the need for further investigation.

References

[1] WANG L, XU G, SHI Y. Rotor airload and acoustics prediction based on CFD/CSD coupling method[J]. Transactions of Nanjing University of Aeronautics and Astronautics, 2018, 35(2): 343-352.

[2] PESKIN C S. The immersed boundary method[J]. Acta Numerica, 2002, 11: 479-517.

[3] MITTAL R, IACCARINO G. Immersed boundary methods[J]. Annual Review of Fluid Mechanics, 2005, 37(1): 239-261.

[4] KIM W, CHOI H. Immersed boundary methods for fluid-structure interaction: A review[J]. International Journal of Heat and Fluid Flow, 2019, 75: 301-309.

[5] VERZICCO R. Immersed boundary methods: Historical perspective and future outlook[J]. Annual Review of Fluid Mechanics, 2023, 55: 129-155.

[6] HUANG Y, ZHANG Y, PU T, et al. A computational framework for parachute inflation based on immersed boundary/finite element approach[J]. Transactions of Nanjing University of Aeronautics and Astronautics, 2024, 41(4): 502-514.

[7] PESKIN C S. Flow patterns around heart valves: A numerical method[J]. Journal of Computational Physics, 1972, 10(2): 252-271.

[8] UDAYKUMAR H S, MITTAL R, RAMPUNGGOON P, et al. A sharp interface Cartesian grid method for simulating flows with complex moving boundaries[J]. Journal of Computational Physics, 2001, 174(1): 345-380.

[9] GILMANOV A, SOTIROPOULOS F. A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies[J]. Journal of Computational Physics, 2005, 207(2): 457-492.

[10] SEO J H, MITTAL R. A sharp-interface immersed boundary method with improved mass conservation and reduced spurious pressure oscillations[J]. Journal of Computational Physics, 2011, 230(19): 7347-7363.

[11] TUCKER P G, PAN Z. A Cartesian cut cell method for incompressible viscous flow[J]. Applied Mathematical Modelling, 2000, 24(8/9): 591-606.

[12] XIE Z. An implicit Cartesian cut-cell method for in-

compressible viscous flows with complex geometries[J]. Computer Methods in Applied Mechanics and Engineering, 2022, 399: 115449.

[13] TSENG Y H, FERZIGER J H. A ghost-cell immersed boundary method for flow in complex geometry[J]. Journal of Computational Physics, 2003, 192(2): 593-623.

[14] CHI C, ABDELSAMIE A, THÉVENIN D. A directional ghost-cell immersed boundary method for incompressible flows[J]. Journal of Computational Physics, 2020, 404: 109122.

[15] TIWARI A, VANKA S P. A ghost fluid Lattice Boltzmann method for complex geometries[J]. International Journal for Numerical Methods in Fluids, 2012, 69(2): 481-498.

[16] LIU C, HU C. A second order ghost fluid method for an interface problem of the Poisson equation[J]. Communications in Computational Physics, 2017, 22(4): 965-996.

[17] PESKIN C S. Numerical analysis of blood flow in the heart[J]. Journal of Computational Physics, 1977, 25(3): 220-252.

[18] ROMA A M, PESKIN C S, BERGER M J. An adaptive version of the immersed boundary method[J]. Journal of Computational Physics, 1999, 153(2): 509-534.

[19] LAI M C, PESKIN C S. An immersed boundary method with formal second-order accuracy and reduced numerical viscosity[J]. Journal of Computational Physics, 2000, 160(2): 705-719.

[20] GOLDSTEIN D, HANDLER R, SIROVICH L. Modeling a No-slip flow boundary with an external force field[J]. Journal of Computational Physics, 1993, 105(2): 354-366.

[21] FADLUN E A, VERZICCO R, ORLANDI P, et al. Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations[J]. Journal of Computational Physics, 2000, 161(1): 35-60.

[22] LIMA E SILVA A L F, SILVEIRA-NETO A, DAMASCENO J J R. Numerical simulation of two-dimensional flows over a circular cylinder using the immersed boundary method[J]. Journal of Computational Physics, 2003, 189(2): 351-370.

[23] UHLMANN M. An immersed boundary method with direct forcing for the simulation of particulate flows[J]. Journal of Computational Physics, 2005, 209(2): 448-476.

[24] FENG Z G, MICHAELIDES E E. The immersed boundary-lattice Boltzmann method for solving fluid-particles interaction problems[J]. Journal of Computational Physics, 2004, 195(2): 602-628.

[25] FENG Z G, MICHAELIDES E E. Proteus: A direct forcing method in the simulations of particulate flows[J]. Journal of Computational Physics, 2005, 202(1): 20-51.

[26] NIU X D, SHU C, CHEW Y T, et al. A momentum exchange-based immersed boundary-lattice Boltzmann method for simulating incompressible viscous flows[J]. Physics Letters A, 2006, 354(3): 173-182.

[27] LUO K, WANG Z, FAN J, et al. Full-scale solutions to particle-laden flows: Multidirect forcing and immersed boundary method[J]. Physical Review E, Statistical, Nonlinear, and Soft Matter Physics, 2007, 76(6): 066709.

[28] WANG Z, FAN J, LUO K. Combined multi-direct forcing and immersed boundary method for simulating flows with moving particles[J]. International Journal of Multiphase Flow, 2008, 34(3): 283-302.

[29] HU Y, YUAN H, SHU S, et al. An improved momentum exchanged-based immersed boundary-lattice Boltzmann method by using an iterative technique[J]. Computers & Mathematics with Applications, 2014, 68(3): 140-155.

[30] CHEN W, ZOU S, CAI Q, et al. An explicit and non-iterative moving-least-squares immersed-boundary method with low boundary velocity error[J]. Journal of Computational Physics, 2023, 474: 111803.

[31] SHU C, LIU N, CHEW Y T. A novel immersed boundary velocity correction-lattice Boltzmann method and its application to simulate flow past a circular cylinder[J]. Journal of Computational Physics, 2007, 226(2): 1607-1622.

[32] WU J, SHU C. Implicit velocity correction-based immersed boundary-lattice Boltzmann method and its applications[J]. Journal of Computational Physics, 2009, 228(6): 1963-1979.

[33] WU J, SHU C. An improved immersed boundary-lattice Boltzmann method for simulating three-dimensional incompressible flows[J]. Journal of Computational Physics, 2010, 229(13): 5022-5042.

[34] DASH S M, LEE T S, HUANG H. A novel flexible forcing hybrid IB-LBM scheme to simulate flow past circular cylinder[J]. International Journal of Modern Physics C, 2014, 25(1): 1340014.

[35] DASH S M, LEE T S, LIM T T, et al. A flexible

forcing three dimension IB-LBM scheme for flow past stationary and moving spheres[J]. Computers & Fluids, 2014, 95: 159-170.

[36] ZHAO X, CHEN Z, YANG L, et al. Efficient boundary condition-enforced immersed boundary method for incompressible flows with moving boundaries[J]. Journal of Computational Physics, 2021, 441: 110425.

[37] DU Y, YANG L, SHU C, et al. Inverse distance weighting interpolation-based immersed boundary velocity correction method for incompressible flows[J]. Physics of Fluids, 2023, 35(8): 083610.

[38] SHEPARD D. A two-dimensional interpolation function for irregularly-spaced data[C]//Proceedings of the 1968 23rd ACM National Conference. [S.l.] : ACM, 1968: 517-524.

[39] FRANKE R. Scattered data interpolation: Tests of some methods[J]. Mathematics of Computation, 1982, 38(157): 181-200.

[40] DU Y, YANG L, XIAO Y, et al. An immersed boundary velocity correction method combined with virtual body-fitted grid for simulation of incompressible flows[J]. Physics of Fluids, 2024, 36(1): 013603.

[41] WU B, DU Y, SHU C. Simplified inverse distance weighting-immersed boundary method for simulation of fluid-structure interaction[J]. Journal of Computational Physics, 2025, 521: 113524.

[42] ZHANG N, ZHENG Z C, ECKELS S. Study of heat-transfer on the surface of a circular cylinder in flow using an immersed-boundary method[J]. International Journal of Heat and Fluid Flow, 2008, 29(6): 1558-1566.

[43] REN W W, SHU C, WU J, et al. Boundary condition-enforced immersed boundary method for thermal flow problems with Dirichlet temperature condition and its applications[J]. Computers & Fluids, 2012, 57: 40-51.

[44] REN W, SHU C, YANG W. An efficient immersed boundary method for thermal flow problems with heat flux boundary conditions[J]. International Journal of Heat and Mass Transfer, 2013, 64: 694-705.

[45] WANG Y, SHU C, YANG L M. Boundary condition-enforced immersed boundary-lattice Boltzmann flux solver for thermal flows with Neumann boundary conditions[J]. Journal of Computational Physics, 2016, 306: 237-252.

[46] DU Y J, YANG L M, SHU C, et al. Virtual body-fitted grid-based immersed boundary method for simulation of thermal flows with Dirichlet and Neumann boundary conditions[J]. Journal of Computational Physics, 2024, 519: 113450.

[47] SHAO J Y, SHU C, CHEW Y T. Development of an immersed boundary-phase field-lattice Boltzmann method for Neumann boundary condition to study contact line dynamics[J]. Journal of Computational Physics, 2013, 234: 8-32.

[48] WANG L, CURRAO G M D, HAN F, et al. An immersed boundary method for fluid-structure interaction with compressible multiphase flows[J]. Journal of Computational Physics, 2017, 346: 131-151.

[49] HUANG W X, GUO W. An improved penalty immersed boundary method for multiphase flow simulation[J]. International Journal for Numerical Methods in Fluids, 2018, 88(9): 447-462.

[50] ROMAN F, ARMENIO V, FRÖHLICH J. A simple wall-layer model for large eddy simulation with immersed boundary method[J]. Physics of Fluids, 2009, 21(10): 101701.

[51] ZHOU C H. RANS simulation of high-*Re* turbulent flows using an immersed boundary method in conjunction with wall modeling[J]. Computers & Fluids, 2017, 143: 73-89.

[52] PU T M, ZHOU C H. An immersed boundary/wall modeling method for RANS simulation of compressible turbulent flows[J]. International Journal for Numerical Methods in Fluids, 2018, 87(5): 217-238.

[53] PARK H, OH G, PARK T S, et al. An immersed boundary formulation incorporating a two-layer wall model approach for RANS simulations with complex geometry[J]. Computers & Fluids, 2020, 205: 104551.

[54] CAI S G, DEGRIGNY J, BOUSSUGE J F, et al. Coupling of turbulence wall models and immersed boundaries on Cartesian grids[J]. Journal of Computational Physics, 2021, 429: 109995.

[55] CONSTANT B, PÉRON S, BEAUGENDRE H, et al. An improved immersed boundary method for turbulent flow simulations on Cartesian grids[J]. Journal of Computational Physics, 2021, 435: 110240.

[56] SHI B, YANG X, JIN G, et al. Wall-modeling for large-eddy simulation of flows around an axisymmetric body using the diffuse-interface immersed boundary method[J]. Applied Mathematics and Mechanics, 2019, 40(3): 305-320.

[57] DU Y, YANG L, SHU C, et al. Wall model-based diffuse-interface immersed boundary method for simu-

lation of incompressible turbulent flows[J]. International Journal for Numerical Methods in Fluids, 2022, 94(11): 1888-1908.

[58] MA M, HUANG W X, XU C X. A dynamic wall model for large eddy simulation of turbulent flow over complex/moving boundaries based on the immersed boundary method[J]. Physics of Fluids, 2019, 31(11): 115101.

[59] MA M, HUANG W X, XU C X, et al. A hybrid immersed boundary/wall-model approach for large-eddy simulation of high-Reynolds-number turbulent flows[J]. International Journal of Heat and Fluid Flow, 2021, 88: 108769.

[60] HUANG W X, CHANG C B, SUNG H J. An improved penalty immersed boundary method for fluid-flexible body interaction[J]. Journal of Computational Physics, 2011, 230(12): 5061-5079.

[61] YAN K, WU Y, ZHU Q, et al. Efficient wall-modeling diffused-interface immersed boundary method for solving turbulent flows with high-order finite difference schemes[J]. Physics of Fluids, 2024, 36(11): 115125.

[62] SPALART P, ALLMARAS S. A one-equation turbulence model for aerodynamic flows[C]//Proceedings of the 30th Aerospace Sciences Meeting and Exhibit. Reno, NV, USA: AIAA, 1992: AIAA1992-439.

[63] MENTER F R. Two-equation eddy-viscosity turbulence models for engineering applications[J]. AIAA Journal, 1994, 32(8): 1598-1605.

[64] SMAGORINSKY J. General circulation experiments with the primitive equations[J]. Monthly Weather Review, 1963, 91(3): 99-164.

[65] TAIRA K, COLONIUS T. The immersed boundary method: A projection approach[J]. Journal of Computational Physics, 2007, 225(2): 2118-2137.

[66] GONCHARUK K, OSHRI O, FELDMAN Y. The immersed boundary method: A SIMPLE approach[J]. Journal of Computational Physics, 2023, 487: 112148.

[67] SHU C, WANG Y, TEO C J, et al. Development of lattice Boltzmann flux solver for simulation of incompressible flows[J]. Advances in Applied Mathematics and Mechanics, 2014, 6(4): 436-460.

[68] LEVEQUE R J, LI Z. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources[J]. SIAM Journal on Numerical Analysis, 1995, 32(5): 1704.

[69] YANG X, ZHANG X, LI Z, et al. A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations[J]. Journal of Computational Physics, 2009, 228(20): 7821-7836.

[70] VANELLA M, BALARAS E. A moving-least-squares reconstruction for embedded-boundary formulations[J]. Journal of Computational Physics, 2009, 228(18): 6617-6628.

[71] WANG X, SHU C, WU J, et al. An efficient boundary condition-implemented immersed boundary-lattice Boltzmann method for simulation of 3D incompressible viscous flows[J]. Computers & Fluids, 2014, 100: 165-175.

[72] IMAMURA T, SUZUKI K, NAKAMURA T, et al. Flow simulation around an airfoil by lattice Boltzmann method on generalized coordinates[J]. AIAA Journal, 2005, 43(9): 1968-1973.

[73] SHU C, REN W W, YANG W M. Novel immersed boundary methods for thermal flow problems[J]. International Journal of Numerical Methods for Heat & Fluid Flow, 2013, 23(1): 124-142.

[74] SUZUKI K, KAWASAKI T, FURUMACHI N, et al. A thermal immersed boundary-lattice Boltzmann method for moving-boundary flows with Dirichlet and Neumann conditions[J]. International Journal of Heat and Mass Transfer, 2018, 121: 1099-1117.

[75] HU Y, LI D, SHU S, et al. An efficient immersed boundary-lattice Boltzmann method for the simulation of thermal flow problems[J]. Communications in Computational Physics, 2016, 20(5): 1210-1257.

[76] GUO T, SHEN E, LU Z, et al. Implicit heat flux correction-based immersed boundary-finite volume method for thermal flows with Neumann boundary conditions[J]. Journal of Computational Physics, 2019, 386: 64-83.

[77] WANG M, MOIN P. Dynamic wall modeling for large-eddy simulation of complex turbulent flows[J]. Physics of Fluids, 2002, 14(7): 2043-2051.

[78] DUPRAT C, BALARAC G, MÉTAIS O, et al. A wall-layer model for large-eddy simulations of turbulent flows with/out pressure gradient[J]. Physics of Fluids, 2011, 23(1): 015101.

[79] LARSSON J, KAWAI S, BODART J, et al. Large eddy simulation with modeled wall-stress: Recent progress and future directions[J]. Mechanical Engineering Reviews, 2016, 3(1): 418.

[80] WILHELM S, JACOB J, SAGAUT P. A new explicit algebraic wall model for LES of turbulent flows under adverse pressure gradient[J]. Flow, Turbu-

lence and Combustion, 2021, 106(1): 1-35.

[81] SPALDING D B. A single formula for the "Law of the wall"[J]. Journal of Applied Mechanics, 1961, 28(3): 455.

[82] SHI B, XU Z, WANG S. A non-equilibrium slip wall model for large-eddy simulation with an immersed boundary method[J]. AIP Advances, 2022, 12(9): 095014.

[83] KRAVCHENKO A G, MOIN P. Numerical studies of flow over a circular cylinder at Re=3 900[J]. Physics of Fluids, 2000, 12(2): 403-417.

[84] PARNAUDEAU P, CARLIER J, HEITZ D, et al. Experimental and numerical studies of the flow over a circular cylinder at Reynolds number 3 900[J]. Physics of Fluids, 2008, 20(8): 085101.

[85] CABOT W, MOIN P. Approximate wall boundary conditions in the large-eddy simulation of high Reynolds number flow[J]. Flow, Turbulence and Combustion, 2000, 63(1): 269-291.

[86] WU J, SHU C. Simulation of three-dimensional flows over moving objects by an improved immersed boundary-lattice Boltzmann method[J]. International Journal for Numerical Methods in Fluids, 2012, 68(8): 977-1004.

[87] YANG L M, SHU C, YANG W M, et al. An immersed boundary-simplified sphere function-based gas kinetic scheme for simulation of 3D incompressible flows[J]. Physics of Fluids, 2017, 29(8): 083605.

[88] WANG Y, SHU C, YANG L M, et al. An immersed boundary-lattice Boltzmann flux solver in a moving frame to study three-dimensional freely falling rigid bodies[J]. Journal of Fluids and Structures, 2017, 68: 444-465.

[89] FERNANDES P C, RISSO F, ERN P, et al. Oscillatory motion and wake instability of freely rising axisymmetric bodies[J]. Journal of Fluid Mechanics, 2007, 573: 479-502.

[90] SHENOY A R, KLEINSTREUER C. Influence of aspect ratio on the dynamics of a freely moving circular disk[J]. Journal of Fluid Mechanics, 2010, 653: 463-487.

[91] LIN X, WU J, ZHANG T. Self-directed propulsion of an unconstrained flapping swimmer at low Reynolds number: Hydrodynamic behaviour and scaling laws[J]. Journal of Fluid Mechanics, 2021, 907: R3.

Acknowledgements This paper was partially supported by the National Natural Science Foundation of China (Nos.92271103, 12202191).

Authors

The first author Prof. **YANG Liming** received his B.S., M.S., and Ph.D. degrees from College of Aerospace Engineering at Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2010, 2013, and 2016, respectively. From 2016 to 2020, he served as a research fellow in Department of Mechanical Engineering at National University of Singapore (NUS). He is currently a professor in College of Aerospace Engineering at NUAA. His research interests focus on computational fluid dynamics and its applications.

The corresponding author Prof. **SHU Chang** received his B.Eng. and M.Eng. degrees from NUAA in 1983 and 1986, respectively, and his Ph.D. degree from the University of Glasgow, UK, in 1991. He is currently a professor in Department of Mechanical Engineering at National University of Singapore (NUS). His primary research interest lies in the development of efficient numerical methods for simulating various fluid flows. To date, he has authored four monographs and published over 380 articles in internationally refereed (SCI-indexed) journals. His work has been cited more than 28 000 times according to Google Scholar. His research interests focus on computational fluid dynamics and its applications.

Author contributions Prof. **YANG Liming** conceived the study, developed the algorithm, conducted the analysis, interpreted the results, and drafted the manuscript. Prof. **SHU Chang** contributed to algorithm development, supported the analysis, and revised the manuscript. Mr. **DU Yinjie** assisted in the development and analysis of the local virtual body-fitted grid-based diffuse-interface immersed boundary method, contributed to data collection and model validation, and co-drafted the manuscript. Prof. **WU Jie** contributed to the research background and supported the development and analysis of the boundary condition-enforced diffuse-interface immersed boundary method. Prof. **WANG Yan** provided critical insights during discussion and led the application of the immersed boundary method to freely falling disks. All authors reviewed and approved the final version of the manuscript.

Competing interests The authors declare no competing interests.

(Production Editor: ZHANG Huangqun)

扩散界面浸入边界法及其应用进展

杨鲤铭[1]，舒　昌[1,2]，杜银杰[1]，吴　杰[1]，王　岩[1]

（1.南京航空航天大学航空学院，南京 210016，中国；2.新加坡国立大学机械工程系，新加坡 117576，新加坡）

摘要：扩散界面浸入边界法（Immersed boundary method，IBM）在模拟复杂几何外形与运动边界的流动问题中已展现出卓越的数值性能。该方法在固定的笛卡尔网格上求解流场，同时将固体边界离散为一系列浸入于流场中的拉格朗日点。边界条件通过在动量方程中引入力项予以实现，浸入边界与流体域之间的耦合则通过插值过程完成。近年来，扩散界面IBM受到了广泛关注，并发展出多种变体，已成功应用于从等温流到热流、从层流到湍流以及从复杂几何外形绕流到流固耦合等多类问题的模拟中。本文首先简要介绍扩散界面IBM的基本原理，随后重点回顾作者团队近年来在该领域的研究进展，最后通过若干复杂动边界问题的应用实例展示该方法优异的数值模拟能力与广泛的适用性。

关键词：浸入边界法；扩散界面；动边界；不可压缩流；湍流