Journal of Computational Physics 230 (2011) 3479–3499

Contents lists available at ScienceDirect

Journal of Computational Physics

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

An immersed boundary method based on discrete stream function formulation for two- and three-dimensional incompressible flows

Shizhao Wang, Xing Zhang *

LNM, Institute of Mechanics, Chinese Academy of Sciences, Beijing 100190, China

ARTICLE INFO

Article history:
Received 7 June 2010
Received in revised form 24 January 2011
Accepted 31 January 2011
Available online 24 February 2011

Keywords:
Immersed boundary method
Discrete stream function formulation
Incompressible flow
Navier–Stokes equations
Parallel computing

ABSTRACT

An immersed boundary method is proposed in the framework of discrete stream function formulation for incompressible flows. In order to impose the non-slip boundary condition, the forcing term is determined implicitly by solving a linear system. The number of unknowns of the linear system is the same as that of the Lagrangian points representing the body surface. Thus the extra cost in force calculation is negligible if compared with that in the basic flow solver. In order to handle three-dimensional flows at moderate Reynolds numbers, a parallelized flow solver based on the present method is developed using the domain decomposition strategy. To verify the accuracy of the immersed-boundary method proposed in this work, flow problems of different complexity (decaying vortices, flows over stationary and oscillating cylinders and a stationary sphere, and flow over low-aspect-ratio flat-plate) are simulated and the results are in good agreement with the experimental or computational data in previously published literatures.

© 2011 Elsevier Inc. All rights reserved.

1. Introduction

The immersed boundary (IB) method is a numerical technique for solving flow problems by removing the mesh conformability condition on the body surfaces [1,2]. This technique has gained popularity recently in the community of computational fluid dynamics due to the great simplification of mesh-generation procedure for complex geometries. Although the IB method was originally aimed at biological flows (laminar and incompressible), now its area of application has expanded to turbulent and supersonic flows.

The IB method was first introduced by Peskin [3] to simulate blood flow interacting with the heart and heart valves. In his IB method, the elastic (fibre-like) boundary was replaced by Lagrangian points connected by springs. Using this technique, the complicated Fluid Structure Interaction (FSI) problem with moving internal boundaries was reformulated as the problem on a regular and stationary fluid domain with an external forcing term [4]. This method was latter extended to handle rigid boundary by increasing the spring rigidity [5] or using a feedback control [6,7]. However, the above methods suffered from very severe time-step restriction due to the stiffness of the system. To avoid this restriction, Mohd-Yusof [8] first proposed to extract the forcing from the numerical solution itself and impose the velocity boundary condition directly. The method was subsequently used by Fadlun et al. [9], Iaccarino and Verzicco [10], and others. This 'direct forcing' IB method can be further categorized into two subtypes: (a) computing the force and imposing the boundary condition through a local velocity reconstruction [8–17]; and (b) evaluating and distributing the force using the regularized delta function [18–21]. In the latter subtype, the usage of a regularized delta function can be regarded as an attribute inherited from the method of Peskin.

* Corresponding author. Tel.: +86 10 82543929; fax: +86 10 82543977.
E-mail address: zhangx@lnm.imech.ac.cn (X. Zhang).

0021-9991/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2011.01.045

Theoretically speaking, IB method can be incorporated into almost any existing numerical schemes for solving flow problems. So far, most successful implementations are in combination with conventional approaches for solving the Navier–Stokes equations, such as the finite difference or finite volume method which is of second-order-accurate in spacial discretization. For the sake of simplicity and efficiency, regular Cartesian grid is usually used in the implementation of IB methods. However, researches are also being in process towards the combination of IB method with other numerical methods, e.g. the finite element method [22], Lattice Boltzmann Methods (LBM) [23] and Smoothed Particle Hydrodynamics (SPH) [24]. IB methods on more complicated grid systems can also been found in the literatures, such as the semi-structured grid with local mesh refinement [25], curvilinear grid [26,27] and even unstructured grid [28,29]. There are also development efforts in combining IB method with high order schemes, such as the spectral method [8] and compact scheme [30,31].

In IB methods for incompressible flows, most frequently used Navier–Stokes formulation is the primitive-variable type. A segregated procedure is usually involved in solving the equations, such as the projection method or SIMPLE-like methods. The stream function–vorticity formulation is seldom used in conjunction with the IB method. Calhoun [32] and Russell and Wang [33] proposed the method that utilized the discontinuity conditions on the stream function and vorticity to impose the non-slip boundary condition. However, these two Cartesian grid methods are conceptually akin to the immersed interface method introduced by LeVeque and Li [34] rather than the immersed boundary method. More recently, Wang et al. [35] presented a IB method based on the vorticity–velocity formulation. However, due to the limitation of the stream function–vorticity formulation itself, the methods above can only be applied to two-dimensional flows. Colonius and Taira [36] proposed a method that combines the IB method with the discrete stream function (null space) approach to solve the Navier–Stokes equation on irregular domains. It should be noted that although this formulation shares some similarities with the stream function–vorticity one, it is valid in both two- and three-dimensional situations.

In this paper, we proposed an alternative implementation of IB method based on the discrete stream function approach. In the method of [36], the standard projection method is reformulated to include the non-slip condition on the Lagrangian points. The projection steps finally result in a Poisson-like equation for the force on these Lagrangian points and a 'nested iteration' is required to solve it. Due to the fact that the size of the linear system to be solved in the inner iteration block is equivalent to that of the original flow problem, the force computing becomes very time-consuming. We simplified this procedure by applying a forcing technique that is similar to that proposed by Su et al. [19]. It significantly reduces both the algorithm complexity and the computational cost. In the present IB method, to determine the force, we only need to solve a small linear system where the number of unknowns is the same as that of the Lagrangian points. Thus the extra computational cost is negligible if compared with that for solving the Navier–Stokes equations. Furthermore, we also explored the utilization of locally refined meshes (with hanging nodes) and domain-decomposed parallel computing to reduce computational time. Some canonical cases are used to test the validity of the forcing strategy proposed in this paper. Numerical results indicate that the method proposed in the paper is sufficiently accurate in imposing the non-slip boundary condition.

The outline of the paper is as follows. The numerical methodology is presented in Section 2 where four subsections are included. In the first subsection, the governing equations are given, followed by a brief introduction of the discrete stream function approach. The third subsection describes the implementation of the IB method that is proposed in this work. The fourth subsection briefly describes the parallel implementation of the present method. Some validation cases are presented in Section 3 in the order of increasing complexity. These testing cases are: the decaying-vortex problem, flow over stationary and oscillating cylinders, flow over a sphere and flow over a low-aspect-ratio flat-plate. Finally, the conclusions are drawn in Section 4.

2. Numerical methodology

2.1. Governing equations

We consider the incompressible Navier–Stokes equations

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) = -\nabla p + \frac{1}{Re}\nabla^2 \mathbf{u} + \mathbf{f}, \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2}$$

where $\mathbf{u}$ is the velocity vector, p the pressure and Re the Reynolds number. $\mathbf{f}$ is the Eulerian body-force that is used to mimic the effects of the immersed body on the flow. The Reynolds number is defined as $Re = UL/\nu$, where U, L and ν are the reference length, reference velocity and kinematic viscosity, respectively.

The immersed boundary is represented by Lagrangian points where the Lagrangian force is defined. The Eulerian and Lagrangian forces are related to each other through a regularized delta function. The mathematical formulation is

$$\mathbf{f}(\mathbf{x}, t) = \int_S \mathbf{F}(\mathbf{X}(s), t)\delta(\mathbf{x} - \mathbf{X}(s))ds, \tag{3}$$

where $\mathbf{x}$ and $\mathbf{X}$ are the position vectors of the Eulerian and Lagrangian points; $\mathbf{f}$ and $\mathbf{F}$ are the Eulerian and Lagrangian forces, respectively.

At the Lagrangian points, the non-slip boundary condition should be satisfied. Using the regularized delta function again, this condition can be expressed as

$$\int_V \mathbf{u}(\mathbf{x},t)\delta(\mathbf{x}-\mathbf{X}(s))d\mathbf{x} = \mathbf{U}_b(\mathbf{X}(s),t), \tag{4}$$

where $\mathbf{U}_b$ is the velocity of the Lagrangian point. In the case of prescribed motions, this velocity is specified as a given value; while in more complicated FSI problems, it is computed by the dynamics of the body.

2.2. Discrete stream function approach

The discrete stream function approach proposed by Perot and coworkers [37–40] is a numerical method for solving incompressible Navier–Stokes equations. This method is valid for both two- and three-dimensional flows and can be applied to both structured and unstructured meshes. As that pointed out in [38], the divergence-free condition is satisfied to machine precision in this method and there are no splitting errors associated with it. It outperforms the classic fractional step methods in computational efficiency. Although it shares some similarities with the stream function–vorticity methods, the discretization procedure and imposition of boundary conditions are quite different.

The discrete stream function approach uses a staggered mesh to discretize the Navier–Stokes equations. The discretized form of Eqs. (1) and (2) can be expressed by a matrix form as

$$\begin{bmatrix} \mathbf{A} & \mathbf{G} \\ \mathbf{D} & \mathbf{0} \end{bmatrix} \begin{bmatrix} q^{n+1} \\ p \end{bmatrix} = \begin{bmatrix} r^n \\ 0 \end{bmatrix} - \begin{bmatrix} bc_1 \\ bc_2 \end{bmatrix} + \begin{bmatrix} \tilde{f} \\ 0 \end{bmatrix}, \tag{5}$$

where q, p, and $\tilde{f}$ are the discrete velocity flux, pressure, and body force, respectively. The discrete velocity u, can be related to q by multiplying the cell face area. **A**, **G** and **D** are the implicit operator, gradient operator and divergence operator respectively. **A** is a square matrix, but **G** and **D** are not. In addition, the negative transpose of the divergence operator is the gradient operator, $\mathbf{G} = -\mathbf{D}^T$. r^n is the explicit right-hand side of the momentum equation. bc_1 and bc_2 are the boundary condition vectors for the momentum and continuity equations.

In the discrete stream function approach, a discrete stream-function s is defined, such that

$$q = \mathbf{C}s, \tag{6}$$

where **C** is the *curl* operator (which is a non-square matrix). This matrix is constructed in such a way that **D** and **C** enjoy the following relation

$$\mathbf{DC} = \mathbf{0}. \tag{7}$$

The definition in Eq. (6) together with the relation in Eq. (7) guarantee the discrete incompressibility. In the discrete stream function approach, another type of curl operator, the *rotational* operator **R**, is also defined such that matrix **R** and matrix **C** enjoy the following relation

$$\mathbf{R} = \mathbf{C}^T. \tag{8}$$

By pre-multiplying the momentum equation with **R**, the pressure can be eliminated from the system. This can be easily seen in the identity equation

$$\mathbf{RG} = -\mathbf{C}^T\mathbf{D}^T = -(\mathbf{DC})^T = 0. \tag{9}$$

Thus the system of (5) is reduced to a single equation for s at each time step

$$\mathbf{C}^T\mathbf{AC}s^{n+1} = \mathbf{R}(r^n - bc_1) + \mathbf{R}\tilde{f} = \mathbf{R}r'^n + \mathbf{R}\tilde{f}. \tag{10}$$

All of the matrices aforementioned (and those in the next subsection) are sparse. They are never explicitly formed but just programmed as operators that perform vector–matrix multiplications. The representations of these operators are given in Appendix A. As to the time advancement, the diffusion term is implicit, the convection term is treated explicitly and a three-step, second-order, low storage, Runge–Kutta scheme is used [40]. The formulation of this scheme is given in Appendix B. We note that no time step superscript is assigned to the forcing term in (10), the update of $\tilde{f}$ will be discussed in the next section.

The matrix $\mathbf{C}^T\mathbf{AC}$ is symmetric, positive-definite and thus can be solved using the Conjugate Gradient (CG) method. A brief introduction of this iterative method, including the pre-conditioner and convergence criterion, is given in Appendix C. After solving (10), the velocity components can be recovered through (6). Although pressure is eliminated in this approach, if it is required, it can still be obtained through a postprocessing step which is independent of the solution procedure for the velocity field.

2.3. Immersed boundary method in framework of discrete stream function formulation

In the work by Colonius and Taira [36], Eq. (10) and the constraint on velocities (non-slip boundary condition) are reformulated into the form of a Karush–Kahn–Tucker (KKT) system; where $\tilde{f}$ appears as a set of Lagrangian multiplier. The LU decomposition (as that in standard fractional step method) is then used to derive an equation for $\tilde{f}$. This equation is of Poisson-type where $(\mathbf{C}^T\mathbf{A}\mathbf{C})^{-1}$ appears in the modified Laplacian operator and a 'nested iteration' is needed in order to solve it.

In this paper, we propose to employ a forcing strategy that is simple and straightforward. The implementation of this strategy is also very easy. Within one step of time advancing (from time index n to $n+1$), this procedure can be summarized as follows:

(a) A 'predicted' stream function is computed with the force components of time step n,

$$\mathbf{RAC}s^* = \mathbf{R}\tilde{r}^n + \mathbf{R}\tilde{f}^n. \tag{11}$$

Here the convection term is treated explicitly and the diffusion is solved implicitly using trapezoidal method. Note that at this stage, this equation is solved only once and no Runge–Kutta sub-steps are performed for the time advancement.

(b) The discrete velocity vectors are reconstructed using the 'predicted' stream function aforementioned,

$$\vec{u}^* = \mathbf{PC}s^*. \tag{12}$$

Here $\mathbf{P}$ is an interpolating operator that is used to construct the velocity vectors at cell centers from its components (scalars) on the face centers.

(c) A force correction is applied to the velocity such that the desired velocity on the boundary is achieved,

$$\frac{\vec{u}^{n+1} - \vec{u}^*}{\Delta t} = \vec{f}'. \tag{13}$$

(d) The forcing term is updated using the correction,

$$\tilde{f}^{n+1} = \tilde{f}^n + \mathbf{Q}\vec{f}'. \tag{14}$$

Here $\mathbf{Q}$ is the an interpolating operator which is used to obtain the scalars at the faces from vectors at cell centers. By construction, $\mathbf{P}$ and $\mathbf{Q}$ enjoy the relation $\mathbf{Q} = \mathbf{P}^T$. The representations of $\mathbf{P}$ and $\mathbf{Q}$ are given in Appendix A.

(e) The stream function at time step $n+1$ is computed with the updated forcing term,

$$\mathbf{RAC}s^{n+1} = \mathbf{R}\tilde{r}^n + \mathbf{R}\tilde{f}^{n+1}. \tag{15}$$

Here a three-step, second-order, low storage Runge–Kutta scheme is used for the time advancement. Within each sub-step of the Runge–Kutta scheme, the diffusion term is solved implicitly using trapezoidal method and the convection is treated explicitly. Note that the forcing term $\tilde{f}$ is fixed in the three sub-steps of the Runge–Kutta scheme.

The procedure of (a)–(e) is repeated until the terminating time is reached.

The crux of this algorithm is to evaluate the force correction vector in Eq. (13) so as to satisfy the constraints on velocities at the Lagrangian points. In this paper, we follow a similar procedure as that in [19] to determine the force correction.

First, both sides of Eq. (13) are interpolated to the Lagrangian point by using a regularized Delta function δ_h,

$$\sum_x \vec{f}'(\mathbf{x})\delta_h(\mathbf{x} - \mathbf{X}_k)h^3 = \frac{\vec{U}^{n+1}(\mathbf{X}_k) - \vec{U}^*(\mathbf{X}_k)}{\Delta t}, \tag{16}$$

where $\vec{U}^{n+1}$ and $\vec{U}^*$ are the desired velocity and predicted velocity at the Lagrangian points respectively. h is the size of the Eulerian grid. $\vec{U}^*$ is evaluated by

$$\vec{U}^*(\mathbf{X}_k) = \sum_x \vec{u}^*(\mathbf{x})\delta_h(\mathbf{x} - \mathbf{X}_k)h^3. \tag{17}$$

Now we define a force correction vector $\vec{F}'$ on the Lagrangian points. It is related to $\vec{f}'$ by

$$\vec{f}'(\mathbf{x}) = \sum_{j=1}^{M} \vec{F}'(\mathbf{X}_\mathbf{j})\delta_h(\mathbf{x} - \mathbf{X}_j)\Delta s, \tag{18}$$

where Δs is the surface area associated with each Lagrangian point. For all cases in this study, Δs is approximately equivalent to h^2. By substituting Eq. (18) into Eq. (16), the equation for $\vec{F}'$ can be then derived and written in the following form

$$\sum_{j=1}^{M}\left(\sum_x \delta_h(\mathbf{x} - \mathbf{X}_j)\delta_h(\mathbf{x} - \mathbf{X}_k)\Delta s h^3\right)\vec{F}'(\mathbf{X}_j) = \frac{\vec{U}^{n+1}(\mathbf{X}_k) - \vec{U}^*(\mathbf{X}_k)}{\Delta t}. \tag{19}$$

From Eq. (19), it is seen that to determine the forces, we only need to solve a small linear system in which the number of unknowns is the same as the number of the Lagrangian points. Thus its computational cost is negligible when comparing

with that in solving the Navier–Stokes equations. After obtaining the force vector at the Lagrangian points, $\vec{f}'$ is then computed by spreading it to the surrounding Eulerian points using Eq. (18).

The regularized Delta function used in the present study is defined as:

$$\delta_h(\mathbf{x}-\mathbf{X}) = \frac{1}{h^3}\phi\left(\frac{x-X}{h}\right)\phi\left(\frac{y-Y}{h}\right)\phi\left(\frac{z-Z}{h}\right). \tag{20}$$

Here ϕ is in the form of a piecewise function that is proposed in [1].

$$\phi(r) = \begin{cases} \frac{1}{8}\left(3-2|r|+\sqrt{1+4|r|-4r^2}\right), & |r| \leqslant 1, \\ \frac{1}{8}\left(5-2|r|-\sqrt{-7+12|r|-4r^2}\right), & 1 \leqslant |r| \leqslant 2, \\ 0, & 2 \leqslant |r|. \end{cases} \tag{21}$$

2.4. Implicit forcing vs. explicit forcing in immersed boundary methods

In variants of immersed boundary method that make use of the delta-function to handle rigid boundaries, the forcing strategy can be categorized into two types: 'implicit' and 'explicit', based on whether a linear system needs to solved to determine the force. The explicit forcing is very straightforward and less time-consuming; while the implicit forcing can enhance the accuracy in imposing the no-slip boundary condition. Discussion on this issue in more detail and more depth is presented in this subsection.

Firstly, we can show that an explicit forcing (such as that proposed in [18]) fails to accurately impose the no-slip boundary condition at time level $n+1$. A brief exposition of this is given as follows. Suppose we compute the Lagrangian force 'explicitly' by (instead of using Eq. (19))

$$\frac{\vec{F}''(\mathbf{X}_k)}{\Delta s} = \frac{\vec{U}^{n+1}(\mathbf{X}_k) - \vec{U}^{*}(\mathbf{X}_k)}{\Delta t}. \tag{22}$$

Similar to Eq. (18), the Eulerian force can be obtained through the following formula

$$\vec{f}''(\mathbf{x}) = \sum_{j=1}^{M} \vec{F}''(\mathbf{X_j})\delta_h(\mathbf{x}-\mathbf{X}_j)\Delta s. \tag{23}$$

This is then followed by a correction step on velocity at Eulerian points

$$\frac{\vec{u}^{**} - \vec{u}^{*}}{\Delta t} = \vec{f}''. \tag{24}$$

By interpolating both sides of Eq. (24) to the Lagrangian points, we can get

$$\begin{aligned} \frac{\vec{U}^{**}(\mathbf{X}_k) - \vec{U}^{*}(\mathbf{X}_k)}{\Delta t} &= \sum_x \vec{f}''\delta_h(\mathbf{x}-\mathbf{X}_k)h^3 = \sum_x \left(\sum_{j=1}^{M}\vec{F}''(\mathbf{X}_j)\delta_h(\mathbf{x}-\mathbf{X}_j)\Delta s\right)\delta_h(\mathbf{x}-\mathbf{X}_k)h^3 \\ &= \sum_{j=1}^{M}\frac{\vec{F}''(\mathbf{X}_j)}{\Delta s}\left(\sum_x \delta_h(\mathbf{x}-\mathbf{X}_j)\delta_h(\mathbf{x}-\mathbf{X}_k)\Delta s^2 h^3\right) \neq \frac{\vec{F}''(\mathbf{X}_k)}{\Delta s}. \end{aligned} \tag{25}$$

Comparing Eq. (25) with Eq. (22), it is easily seen that

$$\vec{U}^{**}(\mathbf{X}_k) \neq \vec{U}^{n+1}(\mathbf{X}_k). \tag{26}$$

Secondly, the non-slip condition can be rigorously imposed at time level $n+1$ only if the entire discretizing procedure for solving the NS equation is taken into consideration in formulating the equation for Lagrangian forces. Examples of such type of forcing can be found in [20] (in the framework of projection method), [36] (in the framework of discrete stream-function approach) and [41] (in the framework of immersed interface method). In this sense, this type of forcing strategy can be termed as 'fully' implicit. However, it suffers from the drawbacks of high complexity in implementation and high cost in computation.

Thirdly, as a compromise between accuracy and cost, a 'partially' implicit forcing (such as Eq. (19)) cannot guarantee the satisfaction of non-slip condition at time level $n+1$ either. Taking the method of Su et al. [19] as an example, the inaccuracy in the boundary condition is incurred by the following two factors: (1) the implicit treatment of the diffusive term in the momentum equation and (2) the pressure correction step in the projection method. In the method proposed in this paper, the sources of inaccuracy are attributed to: (1) the implicit nature of the equation for stream-function and (2) the multi-step time advancement (i.e. the 3-step Runge–Kutta scheme). It should be noted that due to the elimination of pressure, no velocity correction step is needed in the present method.

Finally, for the explicit or 'partially' implicit forcing, the inaccuracy in boundary condition can be rectified through an iterative procedure. Recently, this idea was implemented in [42] as an improved version of the IB method of Uhlmann [18]. Here

a similar solution is proposed in the framework of the present scheme. In between steps (c) and (e) in Subsection 2.3, extra 'force for correction' can be added iteratively until the desired accuracy of boundary condition is achieved. In this 'inner' iteration, the only modification to the original scheme is that the 'predicted' velocity $\vec{u}^*$ in Eq. (13) is now replaced by a 'tentative' velocity at time level $n+1$. It is seen that this iterative scheme is computationally quite expensive. Within a single iteration, Eq. (15) for the stream function is solved thrice; the linear system for the Lagrangian forces equation (19) has to be solved once; furthermore, both the velocity interpolation (Eq. (17)) and force spreading (Eq. (18)) have to be performed once. In Section 3.2.2, the iterative scheme will be further investigated using a real simulation – laminar flow over a stationary cylinder. The boundary condition errors associated with the forcing strategy will be quantified. The necessity and effectiveness of the proposed iterative scheme are also discussed.

2.5. Local mesh refinement and hanging nodes

Despite the fact that the use of Cartesian grid in conjunction with immersed boundary method has many attractive features, the limitation of this combination is also evident. Either uniform or stretched grid can results in a waste of mesh points in the far-field where coarse resolution is sufficient. To reduce the total mesh number while keeping a fine resolution near the immersed boundary, a locally-refined mesh is highly desirable in immersed boundary solvers. One notable feature of the locally-refined mesh is the presence of cells with hanging nodes. The locally-refined mesh can be handled using a hierarchical tree structure or a fully unstructured approach. It is obvious that a code implemented on a structured mesh (with tree structure) executes faster and requires less memory than that on an unstructured mesh [43]. In the present work, the unstructured approach is adopted for the following reasons: (1) cells with hanging nodes are inherently supported (without any special treatment) on an unstructured mesh in the framework of the discrete stream function approach; (2) it easily allows an anisotropic mesh refinement; and (3) a load-balanced mesh partitioning for parallel computing is easily achievable.

In the present method, the cells around hanging nodes as treated as polygons (2D) or polyhedrons (3D). Fig. 1 shows one example of irregular dual cells near a hanging node in a 2D locally-refined mesh. Although the degradation of accuracy is found locally at the hanging nodes, since the number of hanging nodes is very small, the overall accuracy is not affected. Furthermore, in practical simulations, the cells with handling nodes are usually located at places that are far away from the regions that we are interested in (e.g. boundary layer or near wake), thus their influence on the final results is quite limited. The effectiveness of the locally-refined mesh in reducing the total mesh points can be clearly manifested in the simulation of flow over a sphere (see Section 3.4.1). In one simulation of $Re = 300$, keeping the same resolution near the sphere, the use of a locally-refined mesh (with hanging nodes) reduces the number of mesh points from nearly 9 million to only 1.2 million.

2.6. Parallel implementation

When dealing with three-dimensional flows in complex geometries, even in the laminar regime where the Reynolds number is in the range of $10^2 - 10^3$, meshes with several million nodes are often required. When the turnover time of the simulation is worthy of concern, feasible computations are still severely limited by current computing power. Thus parallel processing is now an indispensable part in the development of an efficient flow solver. In this work, the code is parallelized using domain decomposition methodology and ported to a computer cluster with distributed-memory architecture. A well-known graph partitioning software, METIS [44], is used to partition the grid in the preprocessing stage. Message Passing Interface (MPI) library routines are used for data communication in a master–slave algorithm that is designed for the code parallelization. The slave processes solve the flow field within the partition. The flow variables at the partition boundaries are exchanged among the neighboring partitions at each time step. The master process performs the input–output, time step control and computation of Lagrangian forces. The technical details regarding the parallel implementation of the present

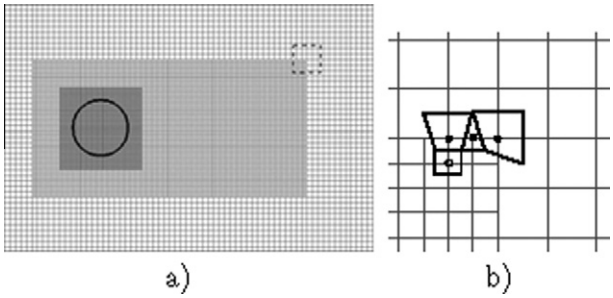

Fig. 1. Example of a 2D locally-refined mesh with hanging-nodes. (a) Mesh used in the simulation of flow over a cylinder. (b) Irregular dual cells near a hanging node. Symbols: open circle – regular node; black square – hanging node; black circle – node on the interface between two different mesh resolutions.

method, such as the data layout for MPI communication, overlapping of communication and computation, 'gathering-and-scattering' strategy for computing Lagrangian forces, and parallel performance, will be summarized and published in a separate paper.

3. Validations

In this section, the validations are arranged in the increasing order of complexity. We first test the order of accuracy of the proposed scheme using the problem of decaying vortices (with an 'artificially' created interior boundary). To validate the method in handling two-dimensional geometries, we then test the solver by simulating laminar flows over a stationary and oscillating cylinder. Finally, the capability of solver in dealing with three-dimensional geometries are tested using laminar flows over a sphere and a low-aspect-ratio flat-plate.

3.1. Decaying vortices: test on the order of accuracy

The exact solution of the decaying vortices problem is given by

$$\begin{aligned} u &= -\cos(\pi x)\sin(\pi y)e^{-\frac{2\pi^2 t}{Re}} \\ v &= \sin(\pi x)\cos(\pi y)e^{-\frac{2\pi^2 t}{Re}} \\ p &= -\frac{1}{4}(\cos(2\pi x)+\cos(2\pi y))e^{-\frac{4\pi^2 t}{Re}} \\ s &= \frac{1}{\pi}\cos(\pi x)\cos(\pi y)e^{-\frac{2\pi^2 t}{Re}}. \end{aligned} \tag{27}$$

This problem is solved numerically on a computational domain ($|x| \leqslant 1.5$, $|y| \leqslant 1.5$). The immersed boundaries are denoted by the thick lines shown in Fig. 2. These lines are produced by rotating a square ($|x| = 1$; $|y| = 1$) 22.5° anticlockwise. The immersed boundaries are then represented by equally spaced Lagrangian points and the desired velocities on these points are set to the exact solutions. The initial velocity condition at $t = 0$ and the velocities at the boundaries of the computational domain ($|x| = 1.5$; $|y| = 1.5$) in time are provided from the exact solution, although only the solution inside the rotated square is of concern. The computations are performed on six uniform quadrilateral meshes with different number of partitions on each side of the computational domain, i.e. 24, 48, 96, 192, 384, and 768. The number of Lagrangian points is adjusted accordingly to ensure that the inter-distance equals the mesh size. In all the computations with different mesh size, the maximum CFL number at $t = 0$ is kept the same as 0.24 by varying the time steps. For all the computations, the Reynolds number Re in Eq. (27) is set to 100. Fig. 3 shows the maximum and L_2-norm of error in u (inside the immersed boundary) at $t = 6.0$ as a function of mesh size h. As shown in this figure, the second-order spacial accuracy of the present method is verified.

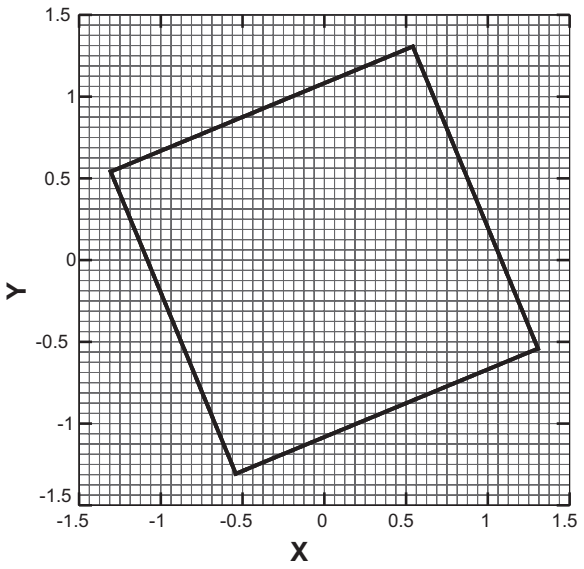

Fig. 2. Computational domain and immersed boundaries for the problem of decaying vortices. The computational domain is ($|x| \leqslant 1.5$, $|y| \leqslant 1.5$). The immersed boundaries denoted by the thick lines are produced by rotating a square ($|x| = 1$; $|y| = 1$) 22.5° anticlockwise.

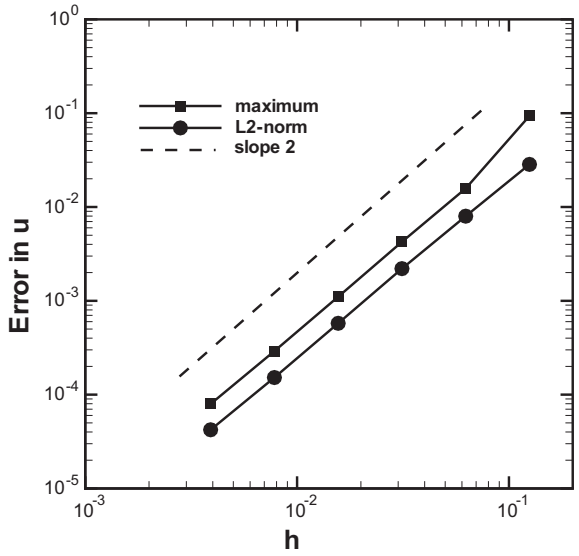

Fig. 3. The maximum and L_2-norm error in u at $t = 6.0$. Only the errors inside the immersed boundaries are calculated.

3.2. Flow over a stationary cylinder

In this section, two-dimensional laminar flows over a stationary cylinder are simulated using the proposed immersed boundary method. Three cases with different Reynolds numbers (based on the free-stream velocity U and diameter D), 40, 100 and 200 are studied.

The simulations are performed in a rectangular domain of $60D \times 40D$. In the case of $Re = 40$, the grid size in the vicinity of the cylinder (a region of $2D \times 2D$) is $0.033D$. For the case of $Re = 100$ and 200, the grid size in the vicinity of the cylinder (a region of $1.5D \times 1.5D$) is $0.02D$. In all three cases, the grids are stretched to the boundaries with an expansion factor of 1.05 and the maximum grid size is $0.5D$. The Lagrangian points are evenly distributed along the circumference of the circular cylinder such that the inter-distance equals the local size of the Eulerian grid approximately.

Uniform free-stream velocity is prescribed at the inlet and the fixed pressure condition is applied at the outlet. On the top and bottom boundaries, slip-wall condition is used. The specification of boundary condition is not very straightforward due to use of s as the primary unknown. For the technical details regarding these boundary conditions in the discrete stream function approach, please refer to the paper by Wang et al. [38]. The non-slip boundary condition on the surface of the cylinder is realized using the immersed boundary method proposed in Section 2.3. The boundary conditions aforementioned are also used in the case of flow over oscillating cylinders in uniform flow (Section 3.3.2) and flow over three-dimensional objects (Section 3.4).

The drag and lift coefficients are computed by

$$C_D = \left(-\sum_{j=1}^{M} F_x(\mathbf{X}_j)\Delta s \right) \Big/ \left(\frac{1}{2}\rho U^2 D \right), \tag{28}$$

$$C_L = \left(-\sum_{j=1}^{M} F_y(\mathbf{X}_j)\Delta s \right) \Big/ \left(\frac{1}{2}\rho U^2 D \right), \tag{29}$$

where F_x and F_y are the Lagrangian forces in the horizontal and vertical direction respectively; U is the free-stream velocity and ρ the density of the fluid.

3.2.1. Flow field and hydrodynamic forces

At $Re = 40$, the flow is steady and a recirculation zone has developed behind the cylinder. The streamlines are shown in Fig. 4. The recirculation zone is characterized by the stream-wise length l, distance from the back of the cylinder to the vortex center a, vertical distance between two vortex centers b, and separation angle θ, as defined in Fig. 4. The results of the present computation and the data from the literatures are listed in Table 1. The characteristic dimensions and drag coefficients computed in the present work are in excellent agreement with the experimental and computational results previously reported in the literatures.

At $Re = 100$ and 200, the flow is characterized by alternating vortex shedding from the upper and lower side of the cylinder. The instantaneous vorticity contours at $Re = 100$ and 200 are shown in Fig. 5. The time history of the drag and lift coefficients for $Re = 100$ and 200 is shown in Fig. 6. The comparisons of the present results (drag and lift coefficients and Strouhal number) with those from the literatures are summarized in Table 2. For both cases, the numerical results from the present simulations are in good agreement with those from the literatures.

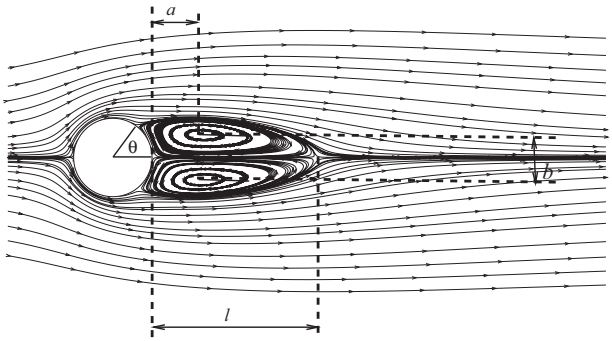

Fig. 4. Streamlines for flow around a cylinder at $Re = 40$. l is the length of the recirculation zone: (a) the stream-wise distance from the cylinder back to the center of one vortex; (b) the gap between the centers of two vortices; θ the separation angle.

Table 1
Drag coefficient and characteristic dimensions of the recirculation zone for flow past a cylinder at $Re = 40$.

Case	l/D	a/D	b/D	θ (°)	C_D
Present	2.36	0.72	0.6	53.8	1.54
Coutanceau and Bouard [45]	2.13	0.76	0.59	53.8	–
Dennis and Chang [46]	2.35	–	–	53.8	1.52
Linnick and Fasel [47]	2.28	0.72	0.6	53.6	1.54
Taira and Colonius [20]	2.33	0.72	0.6	53.7	1.54

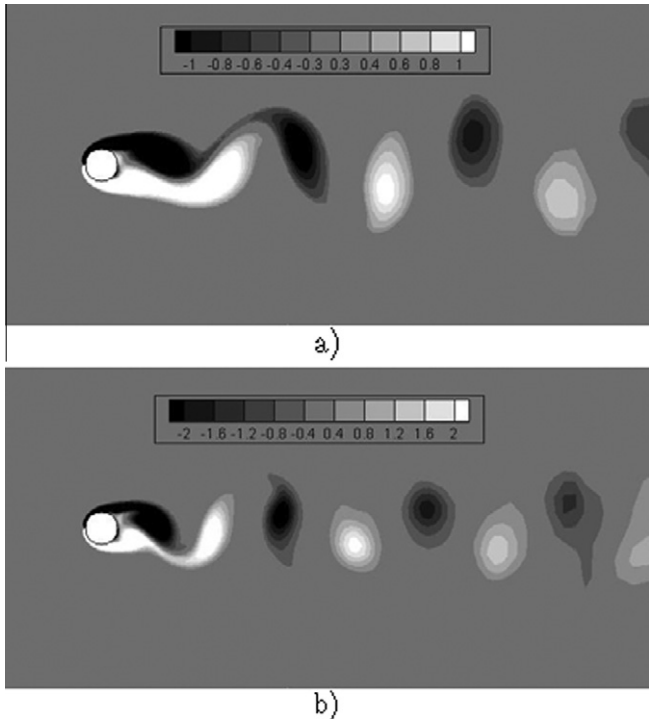

Fig. 5. Instantaneous vorticity contours for flow around a cylinder at (a) $Re = 100$ and (b) $Re = 200$.

3.2.2. Inaccuracy in non-slip boundary condition

According to the discussions in Section 2.4, the forcing strategy proposed in Section 2.3 can lead to inaccurate boundary condition at time level $n + 1$. In this subsection, the errors at the boundary are quantified in the real simulations of flow over a stationary cylinder. We choose the unsteady case of $Re = 100$ to study the deviation of velocity from the value of zero on the Lagrangian points. Through numerical experiments, it is found that the occurrence of the largest deviation coincides with the

phase of zero lift in a vortex-shedding period. Thus the L_2-norm velocity errors at the Lagrangian points are computed at this phase for all testing cases.

First, we perform a test on the variation of errors with the grid sizes. Four different grid sizes ranging from 0.005 to 0.04 are employed in the test. The time step for each case is chosen such that the nominal CFL number is fixed to 0.25 (i.e. $\Delta t = 0.25h/U_\infty$). From the results shown in Fig. 7, the convergence slope of one is clearly seen.

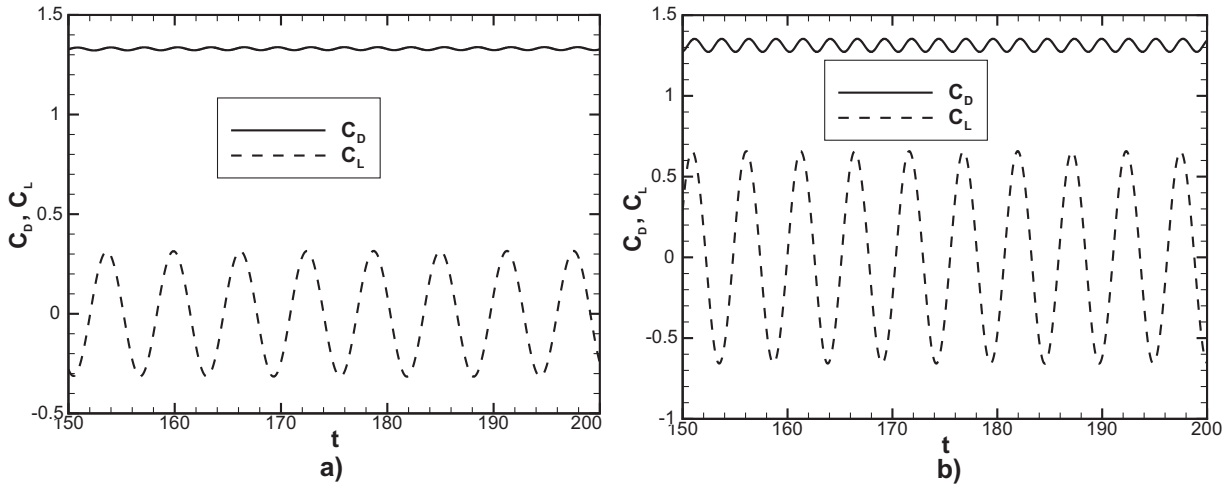

Fig. 6. Time history of C_D and C_L for flow around a cylinder at (a) $Re = 100$ and (b) $Re = 200$.

Table 2
Drag, lift coefficients and Strouhal numbers for flow past a cylinder at $Re = 100$ and $Re = 200$.

Case	*Re*	C_D	C_L	*St*
Present	100	1.33	±0.32	0.166
Liu et al. [48]	100	1.35	±0.32	0.164
Park et al. [49]	100	1.33	±0.33	0.165
Uhlmann [18]	100	1.45	±0.34	0.169
Present	200	1.32	±0.69	0.198
Linnick and Fasel [47]	200	1.34	±0.69	0.197
Liu et al. [48]	200	1.31	±0.69	0.192
Taira and Colonius [20]	200	1.35	±0.68	0.196

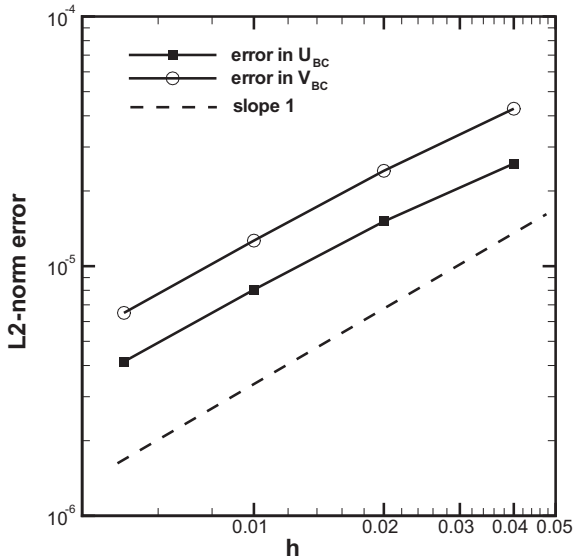

Fig. 7. L_2-norm velocity errors at the Lagrangian points on different grid sizes.

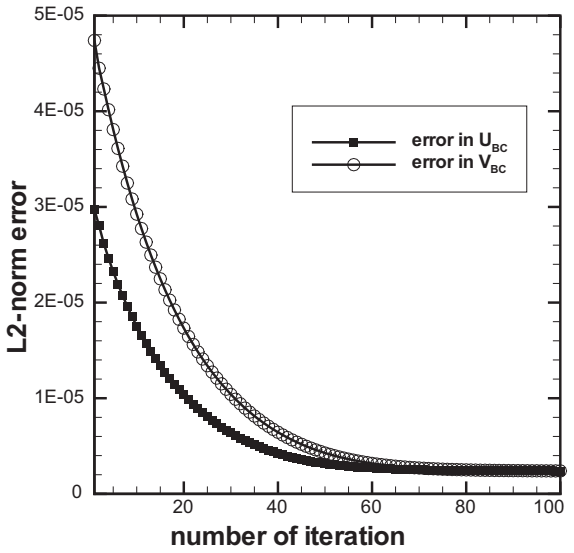

Fig. 8. L_2-norm velocity error at the Lagrangian points as a function of iteration numbers.

Next, we implement the iterative scheme proposed in Section 2.4 in the case of $h = 0.02$ (this is the mesh resolution we use to compute the drag and lift in the previous subsection). Fig. 8 shows the L_2-norm velocity error at the Lagrangian points as a function of iteration numbers. It is seen that the error can be reduced by one order after 50 iterations. All tests indicate that the influences of this rectification measure on the shedding frequency, lift and drag are negligible. On this mesh resolution, the error in the non-slip boundary condition is already sufficiently small (of the order of 10^{-5}) without the iterations. Although the iterative scheme is helpful in reducing the error associated with the boundary condition, it is not recommended due to its high cost.

3.3. Flow over oscillating cylinder

In this subsection, we simulate the flows with in-line and cross-wise oscillating cylinders. They represent two typical cases when the directions of the moving boundary and that of the overall flow are parallel or perpendicular to each other. These two cases are aimed at testing the solver's capability in handling two-dimensional moving boundaries.

3.3.1. In-line oscillating cylinder in fluid at rest

The motion of the in-line oscillating cylinder is prescribed as

$$x(t) = -A\sin(2\pi ft), \tag{30}$$

where A and f are the amplitude and frequency of the oscillation respectively. Using the maximum oscillating velocity $|U|_{\max} = 2\pi Af$ and the diameter of the cylinder D as the reference variables, the dimensionless form of Eq. (30) can be written as

$$\bar{x}(\bar{t}) = -\overline{A}\sin(\bar{t}/\overline{A}), \tag{31}$$

where the barred quantities denote dimensionless variables.

The computational domain for this test is $20D \times 20D$, with the minimum mesh size of $0.025D$ in the vicinity of the cylinder (a region of $4D \times 8D$). The number of Lagrangian points representing the immersed cylinder is 110. In the present simulation, the Reynolds number based on the maximum oscillating velocity and the diameter of the cylinder, $Re = (U_{max}D)/\nu$, is 100. The Keulegan–Carpenter number, based on the maximum oscillating velocity and oscillating frequency, $KC = U_{max}/(fD)$, is 5.0. The dimensionless amplitude corresponding to these two parameters is $\overline{A} = 5/(2\pi)$. A time step is chosen such that the maximum CFL number is 0.5. The non-slip boundary condition is applied to all the side-walls (where the velocity is zero) and also the surface of the cylinder (where the velocity is prescribed by taking the time derivative of Eq. (31)).

Fig. 9 shows the vorticity contours at four different phases, $\bar{t}/\overline{A} = 0,\ \frac{1}{2}\pi,\ \pi$, and $\frac{3}{2}\pi$, respectively. A periodic vortex shedding is clearly seen in these pictures. (Note that the flow field inside the cylinder computed by the present IB method is not shown for clarity.) At this Reynolds number and KC number, the flow pattern that is observed in the simulation is very similar to that in the experimental study of Dutsch et al. [50] and that in the numerical study of Yang and Balaras [14]. Fig. 10 shows the computed profiles of stream-wise and cross-wise velocity components at four different x locations ($x = -0.6D$, $0D$, $0.6D$ and $1.2D$) and three different phases ($\bar{t}/\overline{A} = \pi,\ \frac{7}{6}\pi$, and $\frac{11}{6}\pi$). The result of the present simulation agrees well with the experimental data in [50].

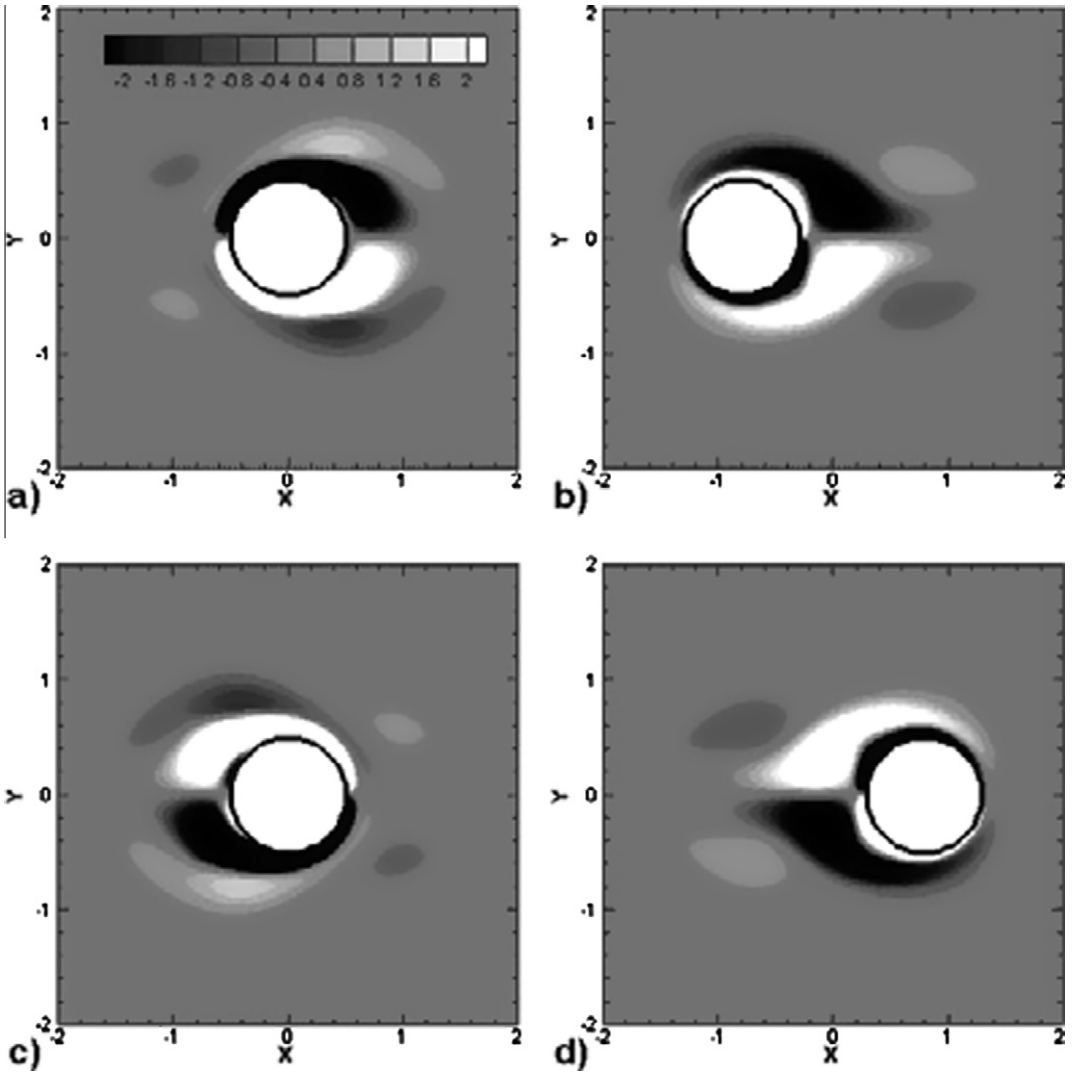

Fig. 9. Instantaneous vorticity contours for an in-line oscillating cylinder in fluid at four different phases: (a) 0; (b) $\frac{1}{2}\pi$; (c) π; and (d) $\frac{3}{2}\pi$.

3.3.2. Cross-wise oscillating cylinder in uniform flow

The motion of the cross-wise oscillating cylinder is prescribed as

$$\bar{y}(\bar{t}) = -\overline{A}sin(2\pi\bar{f}\bar{t}). \tag{32}$$

Here $\bar{f}$ is the dimensionless frequency defined as $\bar{f} = St = (fD)/U$, where D is the diameter of the cylinder and U the velocity of the uniform flow. $\overline{A}$ is the oscillating amplitude non-dimensionalized by D.

The computational domain is $30D \times 20D$. The distance between the center of the cylinder and the inlet is $10D$. The mesh size is $0.025D$ in the vicinity of the cylinder (a region of $4D \times 8D$). The number of the Lagrangian points representing the immersed cylinder is 110. In the present simulation, the Reynolds number based on the free stream velocity and the diameter of the cylinder, $Re = (UD)/\nu$, is 200. The dimensionless oscillating amplitude is $\overline{A} = 0.15$. The selected oscillating St number is 0.198, which is very close to the St number of vortex shedding from a stationary cylinder at $Re = 200$. A time step is chosen such that the maximum CFL number is 0.5.

Fig. 11 shows the instantaneous vorticity contours after the periodicity of the flow is fully established. The von Karman vortex street can be clearly seen in this figure. (Notice that the flow field inside the cylinder computed by the present IB method is not shown for clarity.) The root mean square (rms) velocity magnitude fluctuation is chosen as the representative quantity of the flow field. This quantity is recorded and compared with the experimental results of Griffin [51]. Fig. 12 shows the rms velocity magnitude fluctuation profiles at two different stream-wise locations. Fig. 13 shows the maximum rms velocity–magnitude fluctuation as a function of stream-wise location x. Fig. 14 shows the rms velocity–magnitude fluctuation at the wake axis ($y = 0$). From these comparisons, it is seen that good agreement has been achieved. Thus the capability of the present method in capturing the unsteady flow features in moving boundary problem is further demonstrated.

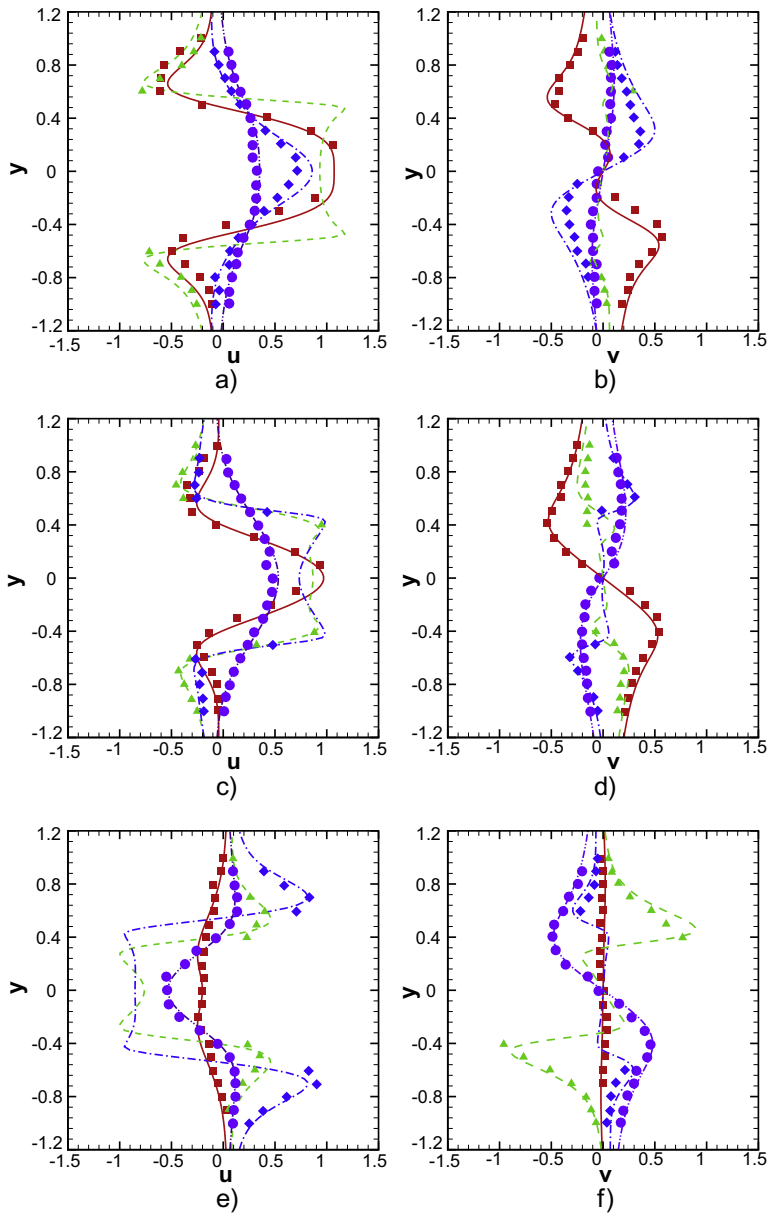

Fig. 10. Computed profiles of u and v respectively at four different x locations and three different phases: (a) and (b) π; (c) and (d) $\frac{7}{6}\pi$; (e) and (f) $\frac{11}{6}\pi$. Lines denote the present results and symbols are the experimental data of Dutsch et al. [50] at: $x = -0.6D$ (red square and red solid line); $x = 0D$ (green delta and green dashed line); $x = 0.6D$ (blue diamond and blue dash-dot line); and $x = 1.2D$ (purple circle and purple dash-dot-dot line).

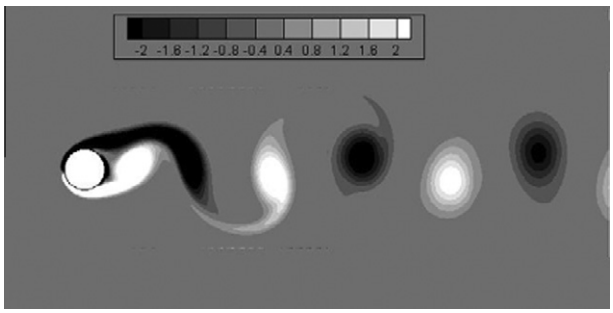

Fig. 11. Instantaneous vorticity contours for flow around a cross-wise oscillating cylinder.

3.4. Three-dimensional flows with immersed bodies

In this subsection, two simulations are performed to test the proposed method in dealing with three-dimensional geometries. Flow over a sphere at low Reynolds numbers is a canonical test. By simulating flow over a low-aspect-ratio flat-plate, we further validate the capability of the method in modeling infinitely thin boundaries.

3.4.1. Flow over a sphere

The simulations of flow over a sphere are performed at $Re = 100$ and 300, where the Reynolds number is based on the free stream velocity U and the diameter of the sphere D. The computation is performed in the domain of $30D \times 30D \times 30D$. For the case of $Re = 100$, a mesh with uniform size of $0.025D$ is deployed in the vicinity of the sphere (a region of $1.5D \times 1.5D \times 1.5D$). For the case of $Re = 300$, a mesh with uniform size of $0.0125D$ is deployed in the vicinity of the sphere (a region of $1.25D \times 1.25D \times 1.25D$). In both cases, the meshes are stretched to the boundaries and the total number of unknowns is

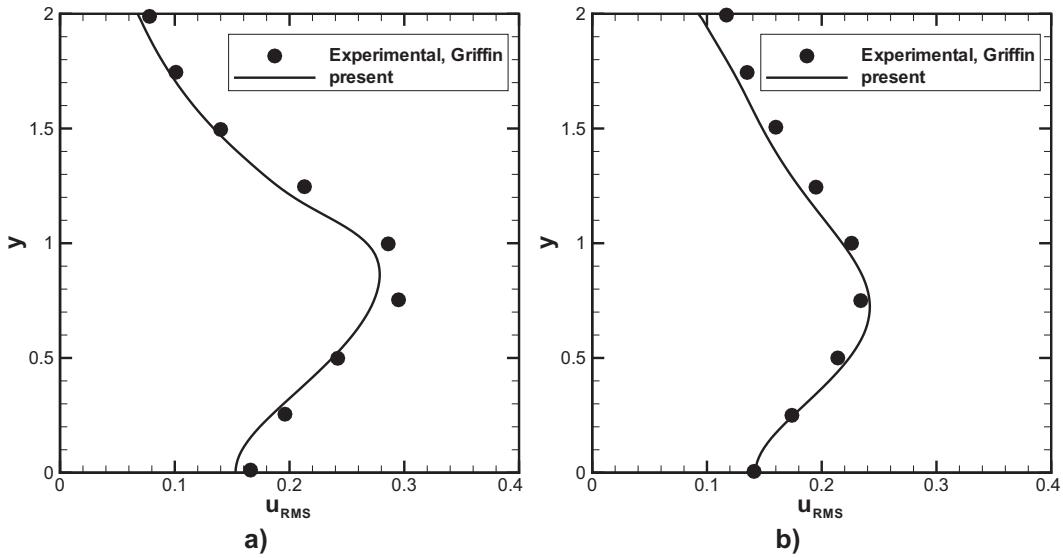

Fig. 12. Computed rms velocity–magnitude fluctuation profiles, with the experimental results of Griffin [51] at two different stream-wise locations: (a) $x = 2.5$ and (b) $x = 5.4$.

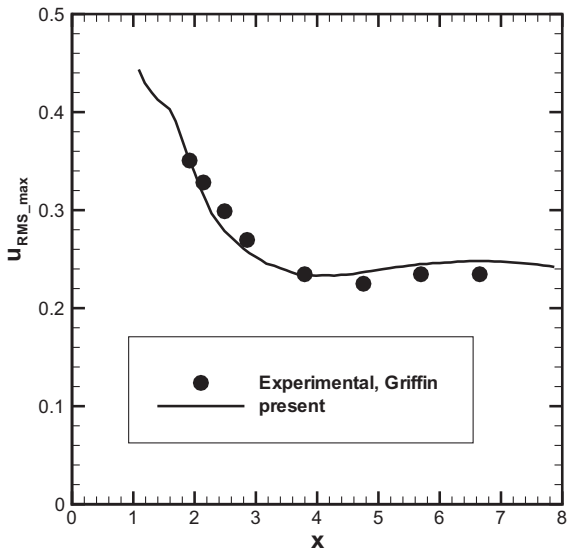

Fig. 13. Maximum rms velocity–magnitude fluctuation vs. stream-wise location x, with the experimental result of Griffin [51].

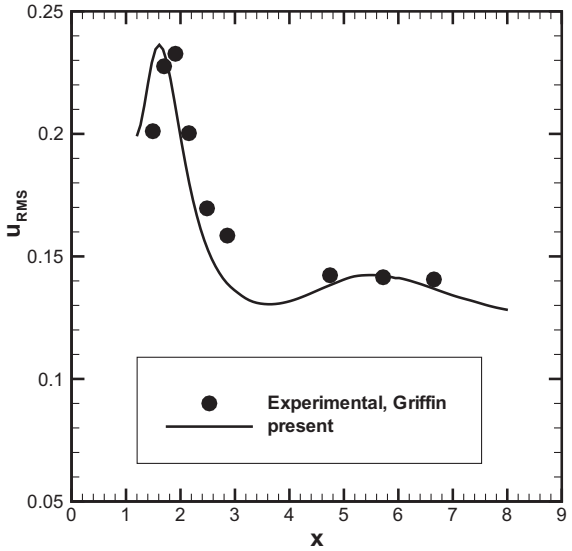

Fig. 14. Computed rms velocity–magnitude fluctuation at $y = 0$ vs. stream-wise location x, with the experimental result of Griffin [51].

Table 3
Drag coefficient for flow past a sphere at $Re = 100$.

Case	C_D
Present	1.13
Johnson and Patel [53]	1.10
Fadlun et al. [9]	1.08

approximately 1 million for the case of $Re = 100$ and around 1.2 million for the $Re = 300$ case. The spherical surface is represented by 4063 and 16,129 Lagrangian points in the two cases respectively. To achieve an almost even distribution of the Lagrangian points on the surface, the method suggested in [18] is used. A uniform velocity is prescribed at the inlet; the fixed pressure condition is applied at the outlet. Due to the large number of grid points required in this case, the simulations are performed on a computer cluster with distributed memory and 128 CPUs are used. The averaged wall-clock time for one step of integration is approximately 3.0 s for the $Re = 100$ case and 4.2 s for the $Re = 300$ case.

The coefficients of drag, lift and side forces are defined as

$$C_D = \left(-\sum_{j=1}^{M} F_x(\mathbf{X}_j)\Delta s \right) \Big/ (0.5\rho U^2 \pi D^2/4), \tag{33}$$

$$C_L = \left(-\sum_{j=1}^{M} F_y(\mathbf{X}_j)\Delta s \right) \Big/ (0.5\rho U^2 \pi D^2/4), \tag{34}$$

$$C_S = \left(-\sum_{j=1}^{M} F_z(\mathbf{X}_j)\Delta s \right) \Big/ (0.5\rho U^2 \pi D^2/4). \tag{35}$$

where F_x, F_y and F_z are the Lagrangian forces in the stream-wise, cross-wise and span-wise direction, respectively.

At $Re = 100$, the flow is steady and axisymmetric with a separation bubble behind the sphere. The drag coefficient of the present calculation is compared with the data from the literatures in Table 3. It is found that a fairly good agreement has been achieved. At $Re = 300$, the flow exhibits unsteady characteristics such as vortex shedding and oscillation in the drag and lift coefficients. A snapshot of vortical structure is shown in Fig. 15, where the vortical surfaces are identified using the method by Jeong and Hussain [52]. This structure of shedding vortices is nearly the same as that in [53]. The averaged drag and lift coefficients are shown in Table 4. It is seen that the present results agree well with those from the literatures.

3.4.2. Flow over a low-aspect-ratio flat-plate

In this subsection, we performed simulations of three-dimensional flow over a low-aspect-ratio flat-plate. We compared our results with the numerical and experimental ones from a recent paper by Taira and Colonius [54]. It is noted that the

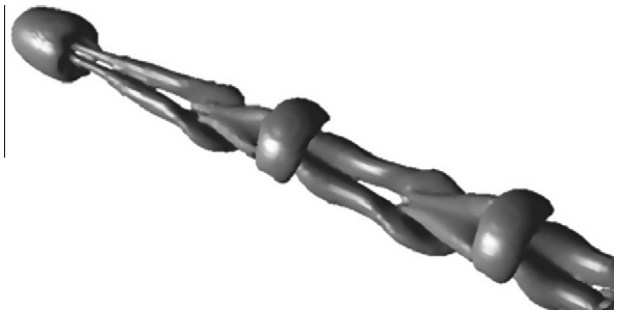

Fig. 15. Instantaneous vortical structure for flow past a sphere at $Re = 300$.

Table 4
Drag, lift coefficients and Strouhal numbers for flow past a sphere at $Re = 300$.

Case	C_D	C_L	St
Present	0.68	0.071	0.135
Johnson and Patel [53]	0.66	0.069	0.137
Kim et al. [11]	0.66	0.067	0.134

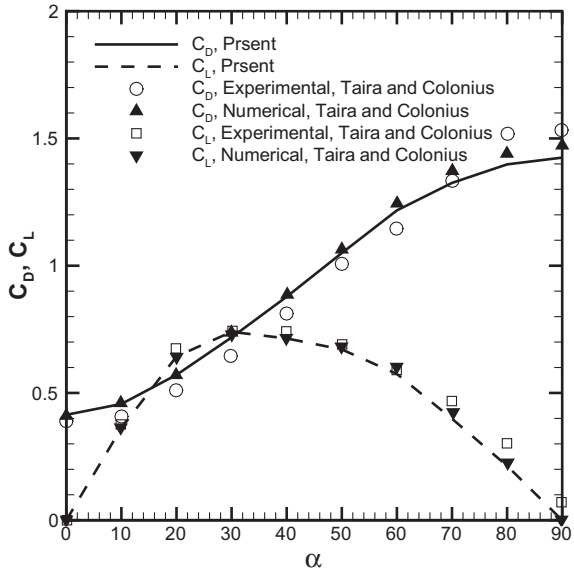

Fig. 16. Variations of lift and drag coefficients with angle of attack α for flow around a flat-plate of AR = 2 and $Re = 100$.

immersed boundary method in the framework of projection method developed in Taira and Colonius [20] was employed in their study.

Our first validation is the steady solution at $Re = 100$ (based on the chord length c). In [54], an experiment was performed in an oil tow-tank (2.4 m × 1.2 m × 1 m). The dimensions of the rectangular plate are 82 mm × 164 mm × 3 mm (AR = 2 with a thickness of $0.037c$). In both the present simulation and that in [54], the flat-plate is modeled as an object of zero thickness (2D surface). In the present study, the computational domain is a box of dimensions $10.1c \times 10c \times 10c$ (which is the same as that in [54]). A locally-refined mesh with the grid number of $150 \times 66 \times 96$ is used in this study. This mesh is generated is such a way that its resolution ($h = 0.025c$) near the rectangular surface is comparable with that of the stretched Cartesian grid in [54]. The spacing of the Lagrangian points distributed on the surface is the same as the local mesh size in the simulation.

The lift and drag coefficients are computed using Eqs. (33) and (34). We compare the lift and drag coefficients at $t = 13$ (after the steady state is reached) with the data in [54]. Fig. 16 shows the comparison of lift and drag coefficients for the angle

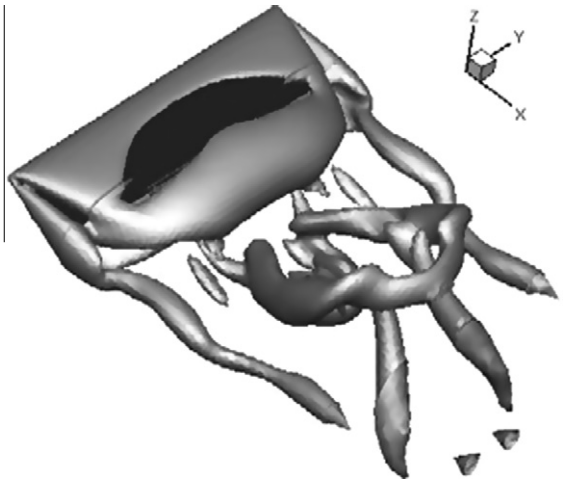

Fig. 17. A snapshot of wake vortex structure behind a flat-plate of AR = 4 at $Re = 300$ and $\alpha = 30^\circ$. Shown is the iso-surface of $Q = 2.0$.

of attack $\alpha \in [0^\circ, 90^\circ]$. The results of the present study are in excellent agreement with the numerical one from [54]. By comparing with the experimental data, it is also found that in both simulations the predictions of lift and drag forces are reasonably good over the full range of α. The slightly larger discrepancy between numerical and experimental results in drag coefficient is attributed to the zero-thickness model used in the computations [54].

Next, we simulate the flow over a rectangular plate with a larger aspect ratio (AR = 4) at $Re = 300$. At the angle of attack of 30°, a periodic solution with vortex shedding is obtained. The non-dimensional frequency (or Strouhal number defined as $St \equiv fc\sin\alpha/U_\infty$) computed in the present study is 0.12, which is the same as the prediction in [54]. Fig. 17 shows one snapshot of vortical structure (characterized by iso-surface of Q-criterion). This pattern is qualitatively consistent with the wake structure shown in [54].

4. Conclusions

In this paper, an immersed boundary method is developed in combination with the discrete stream function formulation for Navier–Stokes equations. The forcing strategy proposed in this work is simple and straightforward and the implementation of this strategy is also very easy. The parallel implementation of the present method facilitates feasible three-dimensional simulations at moderate Reynolds numbers if the turnover time is worthy of concern. The code is validated by various problems of different complexity, such as the problem of decaying vortices, flows over stationary and oscillating cylinders, flow over a sphere and flow over a low-aspect-ratio flat-plate. Good agreement is found between the results in the present work and those in the literatures. This provides a strong evidence of the capability and reliability of this newly proposed method.

Acknowledgements

This work was supported by Chinese Academy of Sciences under the Innovative Project Nos. KJCX-SW-L08 and KJCX3-SYW-S01, National Basic Research Program of China (973 Program) under Project No. 2007CB814800, and National Natural Science Foundation of China under Project Nos. 10702074 and 10872201. The authors also like to thank the Supercomputing Center of Chinese Academy of Sciences (SCCAS) and Shanghai Supercomputer Center (SSC) for the allocation of computing time.

Appendix A

The expressions for the operators used in the discrete stream function approach are given here. The definition of topological entities is shown in Fig. A.1, for both two- and three-dimensional staggered grid system.

To make the expressions clear and concise, the definitions of two unit vectors and two sign conventions are first introduced. $\vec{t}_e$ is the vector pointing from one node to the other sharing the same edge; similarly, $\vec{n}_f$ is the face normal vector pointing from one cell to the other sharing the same face (see Fig. A.1).

The sign convention at each face-cell link is defined as:

$$sign(c,f) = \begin{cases} +1 & c = c_2 \\ -1 & c = c_1. \end{cases} \tag{A.1}$$

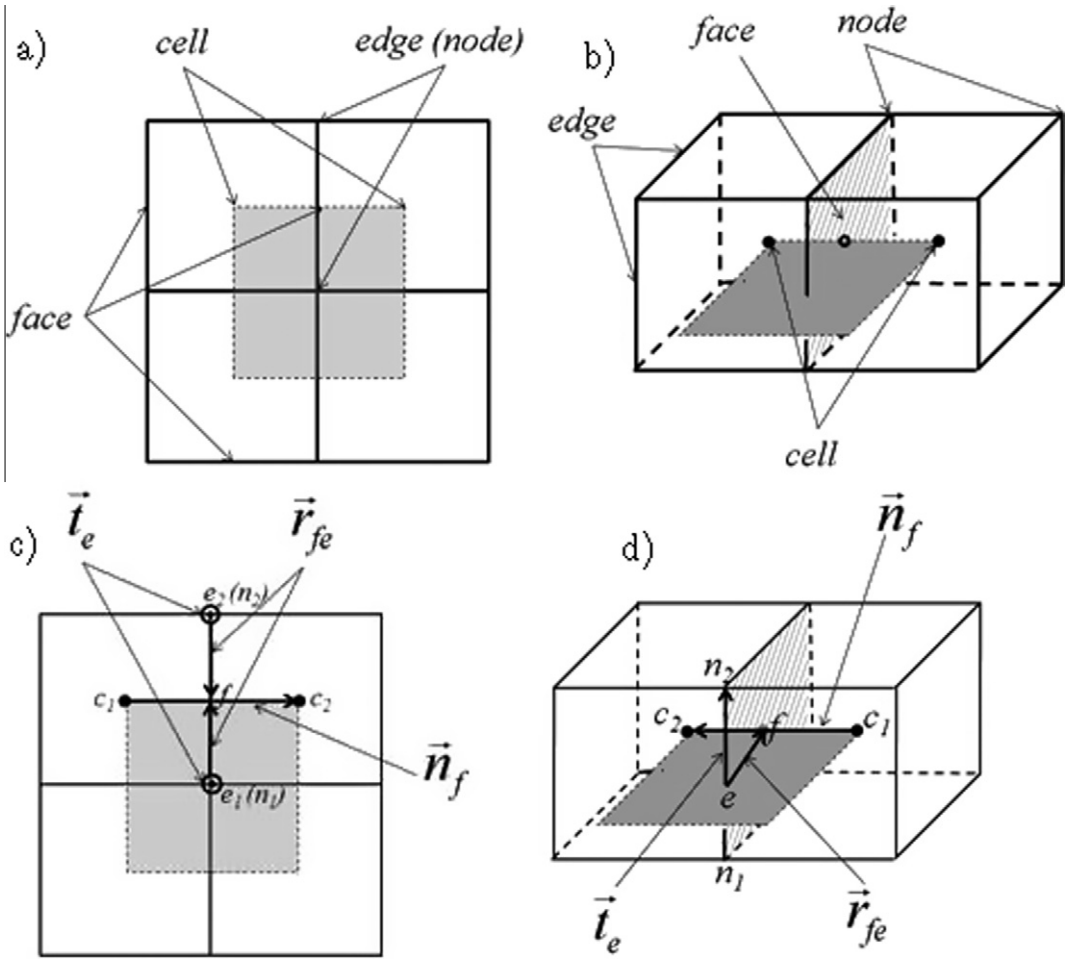

Fig. A.1. The definition of topological entities and sign conventions on a staggered mesh system: (a) topological entities on a 2D mesh; (b) topological entities on a 3D mesh; (c) sign conventions on a 2D mesh; and (d)sign conventions on a 3D mesh.

The sign convention at each edge-face link is defined as:

$$sign(f,e) = sign\left[\vec{n}_f \cdot (\vec{t}_e \times \vec{r}_{fe})\right], \tag{A.2}$$

where $\vec{r}_{fe}$ is the vector pointing from the edge center to the face center.

The normal velocity flux is defined at faces, while the velocity vector is defined at cells. The stream function component s, which is defined at edges, actually represents an integral $l_e(\vec{\psi} \cdot \vec{t}_e)$, where $\vec{\psi}$ is the stream function vector and l_e is the edge length. In two dimensions, the stream function points out of the two-dimensional plane and is located at the edges (or nodes). The schematic representation of variable locations is shown in Fig. A.2.

The discrete gradient, divergence and curl operators can be expressed as

$$\mathbf{G}(p)|_{c\to f} = \sum_{c}^{\substack{face\\cells}} sign(c,f)p_c, \tag{A.3}$$

$$\mathbf{D}(q)|_{f\to c} = \sum_{f}^{\substack{cell\\faces}} sign(c,f)q_f, \tag{A.4}$$

$$\mathbf{C}(s)|_{e\to f} = \sum_{e}^{\substack{face\\edges}} sign(f,e)s_e, \tag{A.5}$$

$$\mathbf{R}(r)|_{f\to e} = \sum_{f}^{\substack{edge\\faces}} sign(f,e)r_f. \tag{A.6}$$

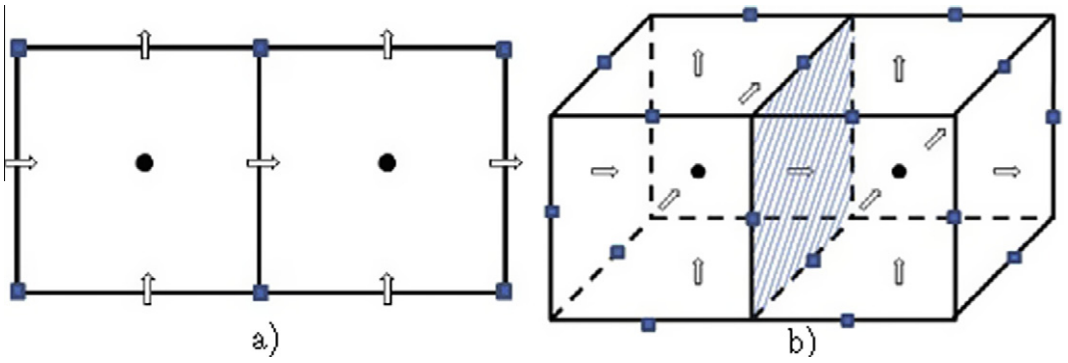

Fig. A.2. The location of variables on a: (a) 2D mesh and (b) 3D mesh. The normal velocity flux q is defined at face centers denoted by ⇒; the stream function component s is defined at edge centers denoted by ■; the velocity vector $\vec{v}$, pressure p and volume force $\vec{f}$ are defined at cell centers denoted by ●.

Here $a \to b$ indicates that an operator manipulates the variable defined at topological entity a and the result is stored at entity b.

The interpolating and integrating operators can be expressed as

$$\mathbf{P}(q)|_{f\to c} = \frac{1}{V_c}\sum_{f}^{\substack{cell\\faces}} q_f \vec{r}_{cf}, \tag{A.7}$$

$$\mathbf{Q}(\vec{u})|_{c\to f} = \sum_{c}^{\substack{face\\cells}} -sign(f,c)(\vec{u}_c \cdot \vec{r}_{cf}), \tag{A.8}$$

where $\vec{r}_{cf}$ is the vector pointing from the cell position to the face position; V_c is the volume of cell.

The implicit operator in Eq. (5) is defined as

$$\mathbf{A} = \frac{\mathbf{I}}{\Delta t} - \frac{1}{2}\frac{1}{Re}\mathbf{L}. \tag{A.9}$$

Here $\mathbf{I}$ is the identity matrix. $\mathbf{L}$ is the Laplacian operator which is defined as

$$\mathbf{L}(q) = \mathbf{Q}\cdot\left\{(1/V_c)\mathbf{D}\cdot\left[(A_f/l_f)\mathbf{G}\cdot\mathbf{P}(q)\right]\right\}, \tag{A.10}$$

where A_f is the area of face and l_f the distance between two cell centers sharing the same face.

Additionally, the convective operator which is used in computing r^n of Eq. (5) can be expressed as

$$\begin{aligned}
&\mathbf{N}(q) = \mathbf{Q}\cdot\left[\left(\frac{1}{V_c}\right)\sum_{f}^{\substack{cell\\faces}}\vec{u}_f q_f\right],\\
&\vec{u}_f = \frac{1}{2}(\vec{u}_{c_1} + \vec{u}_{c_2}),\\
&\vec{u}_c = \mathbf{P}\cdot\mathbf{C}(s).
\end{aligned} \tag{A.11}$$

Appendix B

The three-step, low storage, Runge–Kutta scheme that is used in this work is:

$$\begin{aligned}
&\phi^{n+1,(1)} = \phi^n + \Delta t F(\phi^n),\\
&\phi^{n+1,(2)} = \phi^n + \tfrac{1}{2}\Delta t\left[F(\phi^n) + F(\phi^{n+1,(1)})\right],\\
&\phi^{n+1} = \phi^n + \tfrac{1}{2}\Delta t\left[F(\phi^n) + F(\phi^{n+1,(2)})\right].
\end{aligned} \tag{B.1}$$

This scheme is of second order accurate and very easy to implement. For wave equation, the stability criterion of this particular scheme is that the CFL number based on the wave speed remains less than 2. This restriction is equivalent to require a match between the temporal and spatial accuracy and not too tight in most cases.

Appendix C

For the linear systems that are obtained as a result of the discretization of Eqs. (11) and (15), the coefficient matrices are symmetric and positive-definite. Additionally, the linear system Eq. (19) for computing the force is also symmetric and positive-definite, providing that the inter-distance among Lagrangian points is roughly the same as the grid size used in the Navier–Stokes solver.

In this work, the Conjugate Gradient (CG) method with a Jacobi pre-conditioner is used as an iterative solver to solve the linear system. In the mathematical formulation, a linear system can be written as

$$\mathbf{A}x = \mathbf{b}. \tag{C.1}$$

To improve the rate of convergence of the iterative method, the coefficient matrix can be preconditioned as

$$\mathbf{M}^{-1}\mathbf{A}x = \mathbf{M}^{-1}\mathbf{b}, \tag{C.2}$$

where $\mathbf{M}^{-1}$ is the left pre-conditioner and in this work $\mathbf{M}$ is simply constructed by

$$\mathbf{M} = \mathrm{diag}(\mathbf{A}). \tag{C.3}$$

The pseudo-code for the preconditioned CG Method is given as follows:

Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
For $i = 1, 2 \ldots$
  Solve $Mz^{(i-1)} = r^{(i-1)}$
  $\rho^{(i-1)} = r^{(i-1)\mathrm{T}} z^{(i-1)}$
  If $i = 1$
    $p^{(1)} = z^{(0)}$
  Else
    $\beta^{(i-1)} = \rho^{(i-1)} / \rho^{(i-2)}$
    $p^{(i)} = z^{(i-1)} + \beta^{(i-1)} p^{(i-1)}$
  Endif
  $q^{(i)} = Ap^{(i)}$
  $\alpha^{(i)} = \rho^{(i-1)} / p^{(i)^{\mathrm{T}}} q^{(i)}$
  $x^{(i)} = x^{(i-1)} + \alpha^{(i)} p^{(i)}$
  $r^{(i)} = r^{(i-1)} - \alpha^{(i)} q^{(i)}$
  Check if convergence, continue if necessary
End

We use the following stopping criterion to check if the iteration converges

$$\|r^{(i)}\|^2 \leqslant \varepsilon_1^2 \|r^{(0)}\|^2 + \varepsilon_2^2 \|x^{(0)}\|^2. \tag{C.4}$$

Double precision arithmetic implementation is used for all the simulations in this work. The values of the two controlling parameters in Eq. (C.4) for solving the equation of s are: $\varepsilon_1 = 10^{-9}$; $\varepsilon_2 = 10^{-10}$; while the values for solving Eq. (19) are: $\varepsilon_1 = 10^{-8}$; $\varepsilon_2 = 10^{-9}$.

References

[1] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.
[2] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (2005) 239–261.
[3] C.S. Peskin, Flow patterns around heart valves: a numerical method, J. Comput. Phys. 10 (1972) 252–271.
[4] L.D. Zhu, C.S. Peskin, Simulation of a flexible flapping filament in a flowing soap film by the immersed boundary method, J. Comput. Phys. 179 (2002) 452–468.
[5] M.C. Lai, C.S. Peskin, An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, J. Comput. Phys. 160 (2000) 705–719.
[6] D. Goldstein, R. Handler, L. Sirovich, Modeling a no-slip flow boundary with an external force field, J. Comput. Phys. 105 (1993) 354–366.
[7] E.M. Saiki, S. Biringen, Spatial simulation of a cylinder in uniform flow: application of a virtual boundary method, J. Comput. Phys. 123 (1996) 450–465.
[8] J. Mohd-Yusof, Combined immersed-boundary/B-spline methods for simulations of flow in complex geometries, CTR Annual Research Briefs, NASA Ames/Stanford University, 1997, pp. 317–327.
[9] E.A. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, J. Comput. Phys. 161 (2000) 35–60.
[10] G. Iaccarino, R. Verzicco, Immersed boundary technique for turbulent flow simulations, Appl. Mech. Rev. 56 (2003) 331–347.
[11] J. Kim, D. Kim, H. Choi, An immersed-boundary finite-volume method for simulations of flow in complex geometries, J. Comput. Phys. 171 (2001) 132–150.
[12] Y.H. Tseng, J.H. Ferziger, A ghost-cell immersed boundary method for flow in complex geometry, J. Comput. Phys. 192 (2000) 593–623.
[13] E. Balaras, Modeling complex boundaries using an external force field on fixed Cartesian grids in Large-eddy simulations, Comput. Fluids 33 (2004) 375–404.

[14] J. Yang, E. Balaras, An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries, J. Comput. Phys. 215 (2006) 12–40.
[15] D. Kim, H. Choi, Immersed boundary method for flow around an arbitrarily moving body, J. Comput. Phys. 212 (2006) 662–680.
[16] N. Zhang, Z.C. Zheng, An improved direct-forcing immersed-boundary method for finite difference applications, J. Comput. Phys. 221 (2007) 250–268.
[17] T. Ikeno, T. Kajishima, Finite-difference immersed boundary method consistent with wall conditions for incompressible turbulent flow simulations, J. Comput. Phys. 226 (2007) 1485–1508.
[18] M. Uhlmann, An immersed boundary method with direct forcing for the simulation of particulate flows, J. Comput. Phys. 209 (2005) 448–476.
[19] S.W. Su, M.C. Lai, C.A. Lin, An immersed boundary technique for simulating complex flows with rigid boundary, Comput. Fluids 36 (2007) 313–324.
[20] K. Taira, T. Colonius, The immersed boundary method: a projection approach, J. Comput. Phys. 225 (2007) 2118–2137.
[21] X.L. Yang, X. Zhang, Z.L. Li, G.W. He, A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations, J. Comput. Phys. 228 (2009) 7821–7836.
[22] X.D. Wang, W.K. Liu, Extended immersed boundary method using FEM and RKPM, Comput. Meth. Appl. M. 193 (2004) 1305–1321.
[23] Z.G. Feng, E.E. Michaelides, The immersed boundary-lattice Boltzmann method for solving fluid–particles interaction problems, J. Comput. Phys. 195 (2004) 602–628.
[24] S.E. Hieber, P. Koumoutsakos, An immersed boundary method for smoothed particle hydrodynamics of self-propelled swimmers, J. Comput. Phys. 227 (2008) 8636–8654.
[25] M.D. de Tullio, P. De Palma, G. Iaccarino, G. Pascazio, M. Napolitano, An immersed boundary method for compressible flows using local grid refinement, J. Comput. Phys. 225 (2007) 2098–2117.
[26] D. You, R. Mittal, M. Wang, P. Moin, Computational methodology for large-eddy simulation of tip-clearance flows, AIAA J. 42 (2004) 271–279.
[27] I. Borazjani, L. Ge, F. Sotiropoulos, Curvilinear immersed boundary method for simulating fluid structure interaction with complex 3D rigid bodies, J. Comput. Phys. 227 (2008) 7587–7620.
[28] F.M. Denaro, F. Sarghini, 2-D transmittal flows simulation by means of the immersed boundary method on unstructured grids, Int. J. Numer. Meth. Fluids 38 (2002) 1133–1157.
[29] D.L. Young, Y.J. Jan, C.L. Chiu, A novel immersed boundary procedure for flow and heat simulations with moving boundary, Comput. Fluids 38 (2009) 1145–1159.
[30] S. Laizet, E. Lamballais, High-order compact schemes for incompressible flows: a simple and efficient method with quasi-spectral accuracy, J. Comput. Phys. 228 (2009) 5989–6015.
[31] Paulo J.S.A. Ferreira de Sousa, Jose C.F. Pereira, J.J. Allen, Two-dimensional compact finite difference immersed boundary method, Int. J. Numer. Meth. Fluids (2009), doi:10.1002/fld.2199.
[32] D. Calhoun, A Cartesian grid method for solving the two-dimensional streamfunction–vorticity equations in irregular regions, J. Comput. Phys. 176 (2002) 231–275.
[33] D. Russell, Z.J. Wang, A cartesian grid method for modeling multiple moving objects in 2D incompressible viscous flow, J. Comput. Phys. 191 (2003) 177–205.
[34] R.J. LeVeque, Z. Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (1994) 1019–1044.
[35] Z.L. Wang, J.R. Fan, K.F. Cen, Immersed boundary method for the simulation of 2D viscous flow based on vorticity–velocity formulations, J. Comput. Phys. 228 (2009) 1504–1520.
[36] T. Colonius, K. Taira, A fast immersed boundary method using a nullspace approach and multi-domain far-field boundary conditions, Comput. Meth. Appl. M. 197 (2008) 2131–2146.
[37] B. Perot, Conservation properties of unstructured staggered mesh schemes, J. Comput. Phys. 159 (2000) 58–89.
[38] C. Wang, F. Giraldo, B. Perot, Analysis of an exact fractional step method, J. Comput. Phys. 180 (2002) 183–199.
[39] X. Zhang, D. Schmidt, B. Perot, Accuracy and conservation properties of a three-dimensional unstructured staggered mesh scheme for fluid dynamics, J. Comput. Phys. 175 (2002) 764–791.
[40] B. Perot, R. Nallapati, A moving unstructured staggered mesh method for the simulation of incompressible free-surface flows, J. Comput. Phys. 184 (2003) 192–214.
[41] D.V. Le, B.C. Khoo, J. Peraire, An immersed interface method for viscous incompressible flows involving rigid and flexible boundaries, J. Comput. Phys. 220 (2006) 109–138.
[42] W.P. Breugem, A combined soft-sphere collision/immersed boundary method for resolved simulations of particulate flows, in: Proceedings of the ASME 2010 Third Joint US–European Fluids Engineering Summer Meeting, Montreal, Quebec, Canada, 1–5 August 2010 (FEDSM-ICNMM2010-30634).
[43] S. Kang, G. Iaccarino, F. Ham, P. Moin, Prediction of wall-pressure fluctuation in turbulent flows with an immersed boundary method, J. Comput. Phys. 228 (2009) 6753–6772.
[44] G. Karypis, V. Kumar, A fast and high quality scheme for partitioning irregular graphs, SIAM J. Sci. Comput. 20 (1999) 259–392.
[45] M. Coutanceau, R. Bouard, Experimental determination of the main features of the viscous flow in the wake of a circular cylinder in uniform translation. Part 1. Steady flow, J. Fluid Mech. 79 (1977) 231–256.
[46] S.C.R. Dennis, G. Chang, Numerical solutions for steady flow past a circular cylinder at Reynolds number up to 100, J. Fluid Mech. 42 (1970) 471–489.
[47] M.N. Linnick, H.F. Fasel, A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains, J. Comput. Phys. 204 (2005) 157–192.
[48] C. Liu, X. Zheng, C. Sung, Preconditioned multigrid methods for unsteady incompressible flows, J. Comput. Phys. 139 (1998) 35–57.
[49] J. Park, K. Kwon, H. Choi, Numerical solutions of flow past a circular cylinder at Reynolds numbers up to 160, KSME Int. J. 12 (1998) 1200–1205.
[50] H. Dutsch, F. Durst, S. Becker, H. Lienhart, Low-Reynolds-number flow around an oscillating circular cylinder at low Keulegan–Carpenter numbers, J. Fluid Mech. 360 (1998) 249–271.
[51] O.M. Griffin, The unsteady wake of an oscillating cylinder at low Reynolds number, J. Appl. Mech. 38 (1971) 729–738.
[52] J. Jeong, F. Hussain, On the identification of a vortex, J. Fluid Mech. 285 (1995) 69–94.
[53] T.A. Johnson, V.C. Patel, Flow past a sphere up to a Reynolds number of 300, J. Fluid Mech. 378 (1999) 19–70.
[54] K. Taira, T. Colonius, Three-dimensional flows around low-aspect-ratio flat-plate wings at low Reynolds numbers, J. Fluid Mech. 623 (2009) 187–207.