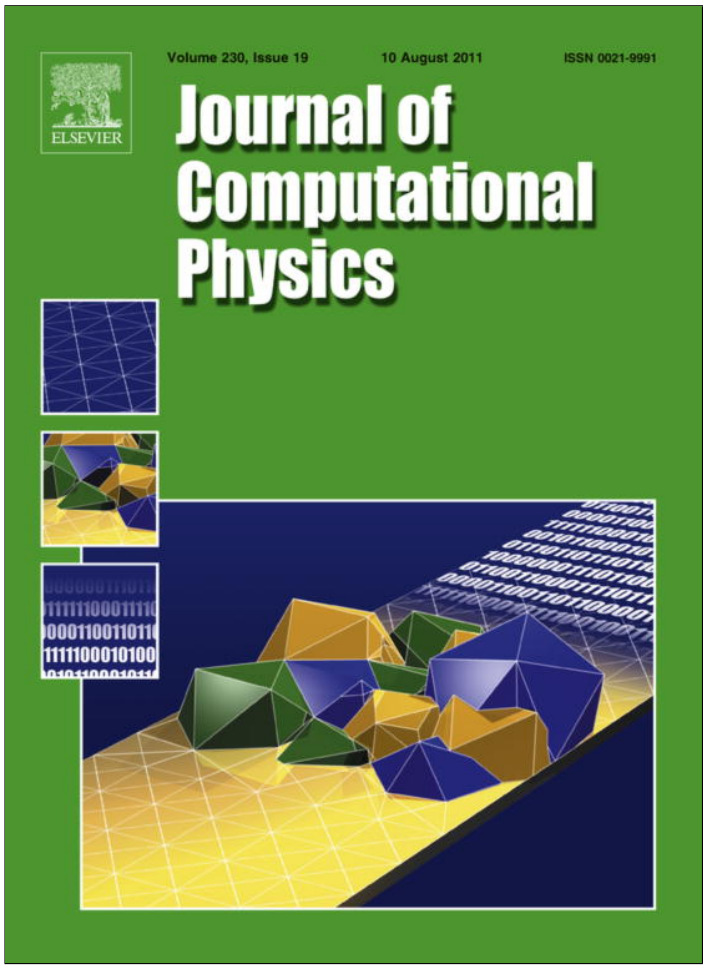

Journal of Computational Physics 230 (2011) 7176–7190

Contents lists available at ScienceDirect

Journal of Computational Physics

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

A boundary condition capturing immersed interface method for 3D rigid objects in a flow

Sheng Xu *

Department of Mathematics, Southern Methodist University, Dallas, TX 75275-0156, USA

ARTICLE INFO

Article history:
Received 24 June 2010
Received in revised form 27 February 2011
Accepted 17 May 2011
Available online 16 June 2011

Keywords:
Boundary condition capturing
The immersed interface method
Flow around 3D moving objects

ABSTRACT

To simulate the flow around an object, we can replace the object with the fluid enclosed by a singular force. We can then simulate the flow on a fixed domain with a fluid–fluid interface supporting the singular force. In this paper, we present a boundary condition capturing approach to determine the singular force for a 3D rigid object. We apply a discontinuous body force to enforce the rigid motion of the fluid replacing the object and compute the singular force based on the kinematics of the object. Due to the singular force and the body force, the flow is not smooth across the interface. We solve the flow using the immersed interface method. Our boundary condition capturing immersed interface method is very efficient and stable, and its accuracy based on the infinity norm is near second order for the velocity and above first order for the pressure.

1. Introduction

There are many Cartesian-grid methods for simulating 3D flows around moving rigid objects [21]. A key element in these methods is to effectively apply the boundary conditions on the objects whose boundaries are not aligned with Cartesian grid lines. According to how these methods do so, Mittal and Iaccarino [21] distinguished them as *discrete* or *continuous* forcing methods, where discrete forcing methods use either *indirect* or *direct* forcing.

In an *indirect discrete* forcing method [40,31,30,5,2,22] boundary conditions are incorporated into flow reconstruction for ghost grid points inside a moving object, and the Navier–Stokes (NS) equations are solved on real and ghost grid points in a time-varying irregular domain. The flow reconstruction modifies discretization schemes near the object with control over the numerical accuracy. However, the linear system after the discretization has different matrix structures near the object and requires robust iterative solvers. In addition, interpolation is needed to assign flow history to so-called "freshly-cleared" grid points.

In a *direct discrete* forcing method [4,12,39,41], boundary conditions are imposed by discrete forcing added in the discrete NS equations, and the discrete equations are solved efficiently on all grid points in a fixed regular domain. Such a method is often combined with projection. Specifically, the forcing is determined to achieve the velocity on an object before the projection, and the flow velocity is then projected on a divergence-free space. The coupling between the forcing and the pressure is often missing or weak. In addition, the nonsmoothness of the velocity near the object is not taken into account in the discretization and the projection, so the accuracy of the velocity near the object is only first order, and the accuracy in the pressure may be worse though it is seldom reported [8].

In a *continuous* forcing method [24,25,6,27,13,17,35], boundary conditions are represented by singular forces added in the original NS equations, and the modified NS equations are then solved on a fixed regular domain with a particular way to

* Tel.: +1 214 768 2985.
E-mail address: sxu@smu.edu
URL: http://faculty.smu.edu/sxu

0021-9991/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2011.05.019

handle the force singularity. By modeling a rigid object with *ad hoc* stiff spring models or feedback controls [6,13,35,36], a continuous forcing method has the stiffness problem [29,23,10], which includes trial-and-error adjustment of free parameters, spurious oscillations near the object, and numerical instability at high Reynolds numbers. Implicit schemes require nonlinear solvers, but they may not always help remove numerical instability [23]. An augmented-variable approach [14,11] determines the singular force for an object by solving the mapping between the singular force and the velocity on the object, which can be made linear in a discrete form. The linear mapping is solved iteratively by GMRES when the object is moving as the explicit construction of the mapping is too expensive. It can be ill-conditioned [11].

We present a boundary-condition capturing immersed interface method with the motivation to avoid some disadvantages of the above methods. Our method is a significant extension of the idea proposed in [37] for 2D flows. It utilizes the necessary new formulas derived in [38] and consists of new ways to implement these formulas. Our method is a continuous forcing method but has numerical stability at high Reynolds numbers under the standard CFL condition, which is in contrast to ad hoc penalty approaches used in [6,13,35,36]. Our method solves the mapping between the singular force and the velocity on an object, but it is different from an augmented-variable approach [14,11] in that our linear mapping is *explicitly* constructed and *directly* solved. The computational efficiency of our method is comparable to a typical direct discrete forcing method [4,12,39,41], but the accuracy near an object is higher (near second order for the velocity and above first order for the pressure). Unlike an indirect discrete forcing method [40,31,30,5,2,22], our method solves a flow on all grid points in a fixed regular domain. It wastes some computations on the grid points inside an object, but it allows for any fast *direct* or *iterative* solvers which are suitable for the simple lid-driven cavity flow. In our method, the effect of moving objects is included only in the right-hand side of the linear system from discretization, and the coefficient matrix of the linear system is exactly the same as that for the cavity flow.

The boundary condition capturing feature of our method can be seen from the perspective of some body-fitted grid methods in which the Dirichlet velocity condition directly enters finite difference discretization, and the following Neumann pressure condition on a moving object is used

$$\frac{\partial p}{\partial n} = -\left(\frac{\nabla \times \vec{\omega}}{Re} + \vec{a}\right) \cdot \vec{n}, \tag{1}$$

where p is the pressure, $\vec{n}$ is the normal to the boundary of the object, $\vec{\omega}$ is the vorticity, Re is the Reynolds number, and $\vec{a}$ is the acceleration of the boundary. In our method, the Dirichlet velocity condition directly enters the calculation of the singular force and the discontinuous body force for an object and indirectly enters finite difference discretization through incorporation of jump conditions induced by these forces, and the Neumann pressure condition is given as

$$\left.\frac{\partial p}{\partial n}\right|_{S^+} = \left[\frac{\partial p}{\partial n}\right] + \left.\frac{\partial p}{\partial n}\right|_{S^-}, \tag{2}$$

where S^+ and S^- denote the fluid and solid side of the boundary, respectively, and $[\cdot]$ denotes a jump condition across the boundary. The forces are determined such that the right-hand side of Eq. (2) is equivalent to that of Eq. (1).

Our boundary condition capturing approach is embedded in the immersed interface method [18]. The immersed interface method was first proposed by LeVeque and Li [15,16] to improve the accuracy of Peskin's immersed boundary method [24–26]. The two methods share the same formulation to represent the effect of an object as a singular force but differ in the numerical treatment of the force singularity. In Peskin's immersed boundary method, the force singularity is regularized by a discretized smooth function, which leads to only first-order accuracy [1,7]. In the immersed interface method, jump conditions induced by the force singularity are incorporated into numerical schemes, which leads to second-order or higher accuracy. The required jump conditions can be derived systematically as in [34]. The incorporation of the jump conditions is based on a generalized Taylor expansion [34]

$$g(s^-_{m+1}) = \sum_{n=0}^{\infty} \frac{g^{(n)}(s^+_0)}{n!}(s_{m+1}-s_0)^n + \sum_{l=1}^{m}\sum_{n=0}^{\infty} \frac{[g^{(n)}(s_l)]}{n!}(s_{m+1}-s_l)^n, \tag{3}$$

where $g(s)$ is a function shown in Fig. 1(a), and $[g^{(n)}(s_l)] = g^{(n)}(s^+_l) - g^{(n)}(s^-_l)$ denotes a jump condition along the s direction. With the stencil shown in Fig. 1(b), second-order standard finite difference and interpolation for a solution $g(s)$ with discontinuities at ξ and η are modified as

$$\begin{aligned}\frac{dg(s^-_i)}{ds} &= \frac{g(s^-_{i+1}) - g(s^+_{i-1})}{2h} + \mathcal{O}(h^2)\\ &\quad + \frac{1}{2h}\left(\sum_{n=0}^{2}\frac{-[g^{(n)}(\xi)]}{n!}(s_{i-1}-\xi)^n - \sum_{n=0}^{2}\frac{[g^{(n)}(\eta)]}{n!}(s_{i+1}-\eta)^n\right),\end{aligned} \tag{4}$$

$$\begin{aligned}\frac{d^2g(s^-_i)}{ds^2} &= \frac{g(s^-_{i+1}) - 2g(s_i) + g(s^+_{i-1})}{h^2} + \mathcal{O}(h^2)\\ &\quad - \frac{1}{h^2}\left(\sum_{n=0}^{3}\frac{-[g^{(n)}(\xi)]}{n!}(s_{i-1}-\xi)^n + \sum_{n=0}^{3}\frac{[g^{(n)}(\eta)]}{n!}(s_{i+1}-\eta)^n\right),\end{aligned} \tag{5}$$

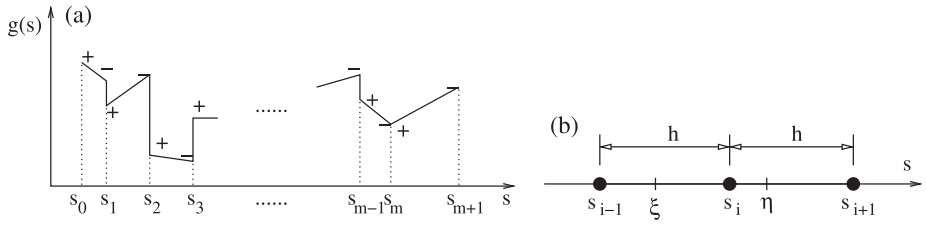

Fig. 1. (a) A piecewise continuous function; (b) a finite difference or interpolation stencil.

$$g(s_i^-) = \frac{g(s_{i-1}^+) + g(s_{i+1}^-)}{2} + \mathcal{O}(h^2) + \frac{1}{2}\left(\left[\frac{\partial g(\xi)}{\partial s}\right](s_{i-1} - \xi) + [g(\xi)]\right) - \frac{1}{2}\left(\left[\frac{\partial g(\eta)}{\partial s}\right](s_{i+1} - \eta) + [g(\eta)]\right). \tag{6}$$

We organize the rest of the paper as follows. In Section 2, we give the formulation used in our boundary condition capturing immersed interface method. In Section 3, we present the numerical implementation of the formulation. In Section 4, we show test results to demonstrate the accuracy, efficiency, and stability of our method. In Section 5, we conclude the paper.

2. Formulation

As shown in Fig. 2(a), we denote the boundary of an object as S and the Cartesian coordinates $\vec{x} = (x, y, z)$ on the boundary as $\vec{X} = (X, Y, Z)$, and we parametrize S by the Lagrangian parameters α_1 and α_2. Tangents and normals to S are calculated as

$$\vec{T} = (T_1, T_2, T_3) = \frac{\partial \vec{X}}{\partial \alpha_1}, \quad J_1 = \|\vec{T}\|_2, \quad \vec{\tau} = \frac{\vec{T}}{J_1}, \tag{7}$$

$$\vec{B} = (B_1, B_2, B_3) = \frac{\partial \vec{X}}{\partial \alpha_2}, \quad J_2 = \|\vec{B}\|_2, \quad \vec{b} = \frac{\vec{B}}{J_2}, \tag{8}$$

$$\vec{N} = (N_1, N_2, N_3) = \vec{T} \times \vec{B}, \quad J = \|\vec{N}\|_2, \quad \vec{n} = \frac{\vec{N}}{J}, \tag{9}$$

where $\vec{\tau}, \vec{b}$ and $\vec{n}$ are the normalized vectors corresponding $\vec{T}, \vec{B}$ and $\vec{N}$, respectively. The parameters α_1 and α_2 are chosen to make $\vec{n}$ point to the outside of the object.

In our boundary condition capturing immersed interface method, the object is treated as the fluid and effectively represented as a singular force and a body force in the incompressible NS equations (multiple objects can be included similarly), which read

$$\frac{\partial \vec{v}}{\partial t} + \nabla \cdot (\vec{v}\vec{v}) = -\nabla p + \frac{1}{Re}\Delta \vec{v} + \vec{q} + \int_S \vec{F}\delta(\vec{x} - \vec{X})\,\mathrm{d}\alpha_1 \mathrm{d}\alpha_2, \tag{10}$$

$$\Delta p = -\left(\frac{\partial D}{\partial t} + \nabla \cdot (2\vec{v}D) - \frac{1}{Re}\Delta D\right) + 2\left(\frac{\partial u}{\partial x}\frac{\partial v}{\partial y} - \frac{\partial u}{\partial y}\frac{\partial v}{\partial x} + \frac{\partial u}{\partial x}\frac{\partial w}{\partial z} - \frac{\partial u}{\partial z}\frac{\partial w}{\partial x} + \frac{\partial v}{\partial y}\frac{\partial w}{\partial z} - \frac{\partial v}{\partial z}\frac{\partial w}{\partial y}\right) + \nabla \cdot \left(\vec{q} + \int_S \vec{F}\delta(\vec{x} - \vec{X})\,\mathrm{d}\alpha_1 \mathrm{d}\alpha_2\right), \tag{11}$$

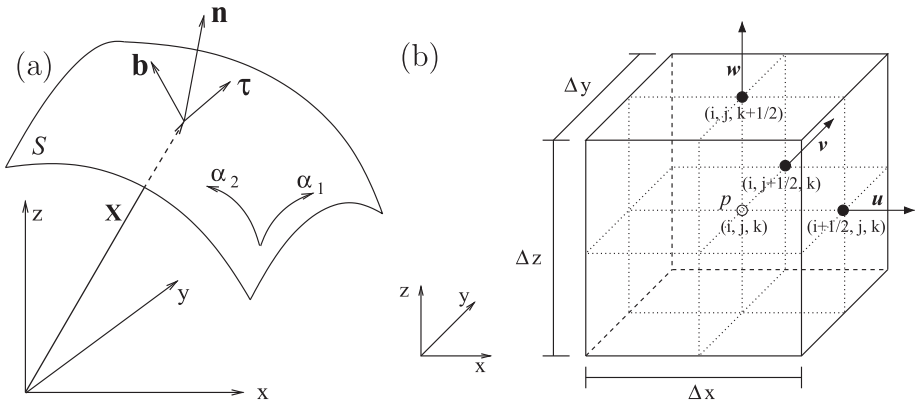

Fig. 2. (a) Geometric description of the boundary of an object; (b) staggered arrangements of the velocity and the pressure on a MAC grid.

where $\vec{v} = (u, v, w)$ is the velocity, p is the pressure, Re is the Reynolds number, $\vec{q}$ is the body force, $\vec{F} = \vec{F}(\alpha_1, \alpha_2, t)$ is the density of the singular force, $\delta(\vec{x} - \vec{X})$ is the 3D Dirac δ function, and $D = \nabla \cdot \vec{v}$ is the divergence of the velocity. We keep the terms with D to better enforce the incompressibility condition. Unless otherwise specified, all variables and quantities are nondimensionalized by the length scale L, the velocity scale U and the fluid density ρ, which define the Reynolds number $Re = \frac{\rho UL}{\mu}$, where μ is the dynamic viscosity of the fluid.

2.1. Body force

Let $\vec{x}$ be the Cartesian coordinates of a point in the rigid object. The velocity of this point is given by

$$\vec{v} = \frac{d\vec{x}}{dt} = \frac{d\vec{x}_C}{dt} + \vec{\Omega} \times (\vec{x} - \vec{x}_C), \tag{12}$$

where $\vec{x}_C$ is the Cartesian coordinates of the center of mass of the object, and $\vec{\Omega}$ is the angular velocity of the object, which can be expressed in terms of three Euler angles [38]. It can be shown that this velocity satisfies

$$\frac{d\vec{v}}{dt} = -\nabla p + \frac{d\vec{\Omega}}{dt} \times (\vec{x} - \vec{x}_C) + \frac{1}{Re}\Delta\vec{v}, \tag{13}$$

where p (subject to a constant) is

$$p = -\frac{d^2\vec{x}_C}{dt^2} \cdot (\vec{x} - \vec{x}_C) + \frac{1}{2}\|\vec{\Omega} \times (\vec{x} - \vec{x}_C)\|_2^2. \tag{14}$$

To make the motion of the object-replacing fluid enclosed by the singular force be the same as the rigid motion of the object, we apply the body force $\vec{q}$ in Eq. (10). According to Eq. (13), the body force applied inside the object-replacing fluid is

$$\vec{q} = \frac{d\vec{\Omega}}{dt} \times (\vec{x} - \vec{x}_C). \tag{15}$$

Outside, $\vec{q}$ has to be zero. The body force $\vec{q}$ is therefore discontinuous with the jump

$$[\vec{q}] = -\frac{d\vec{\Omega}}{dt} \times (\vec{X} - \vec{x}_C). \tag{16}$$

Hereafter $[\cdot] = (\cdot)_{S^+} - (\cdot)_{S^-} = (\cdot)_{\vec{n}^+} - (\cdot)_{\vec{n}^-}$ denotes a jump condition across the boundary S along its normal direction $\vec{n}$.

In each time step, we can advance the momentum equation, Eq. (10), without the body force and then set the inside velocity to the desired one given by Eq. (12), which is equivalent to applying the body force $\vec{q}$ in the momentum equation. An efficient way to determine if a grid point falls inside the object is given in Appendix A.

2.2. Singular force

Let $\vec{f}, f_n$ and $\vec{f}_{\tau b}$ be the force density and its normal and tangential components in the Cartesian (x,y,z) space, and $\tilde{f}_1$ and $\tilde{f}_2$ be its contravariant components in the parameter (α_1, α_2) space. We have

$$\vec{f} = \frac{\vec{F}}{J}, \quad f_n = \vec{f} \cdot \vec{n}, \quad \vec{f}_{\tau b} = \vec{f} - f_n\vec{n}, \tag{17}$$

$$\tilde{f}_1 = (\vec{B} \times \vec{n}) \cdot \vec{f}_{\tau b}, \quad \tilde{f}_2 = (\vec{n} \times \vec{T}) \cdot \vec{f}_{\tau b}. \tag{18}$$

The tangential force $\vec{f}_{\tau b}$ is related to the jump condition of the normal derivative of the velocity as the following [34]

$$\vec{f}_{\tau b} = -\frac{1}{Re}\left[\frac{\partial\vec{v}}{\partial n}\right]. \tag{19}$$

From Eq. (12), we have

$$\left.\frac{\partial\vec{v}}{\partial n}\right|_{S^-} = \vec{\Omega} \times \vec{n}. \tag{20}$$

We can then determine $\vec{f}_{\tau b}$ from

$$\vec{f}_{\tau b} = -\frac{1}{Re}\left(\left.\frac{\partial\vec{v}}{\partial n}\right|_{S^+} - \vec{\Omega} \times \vec{n}\right). \tag{21}$$

We need $\left.\frac{\partial\vec{v}}{\partial n}\right|_{S^+}$ to compute $\vec{f}_{\tau b}$, which can be approximated *explicitly* using a one-sided finite difference such as

$$\frac{\partial\vec{v}(S_0)}{\partial n} = \frac{-3\vec{v}(S_0) + 4\vec{v}(S_1) - \vec{v}(S_2)}{2\Delta n} + O(\Delta n^2), \tag{22}$$

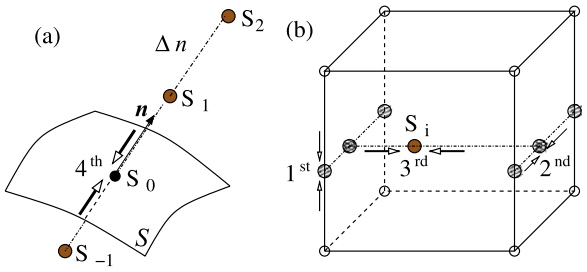

Fig. 3. (a) A finite difference or interpolation stencil; (b) a trilinear interpolation cell (marked with the interpolation turn).

where Δn is the spacing of the finite difference stencil shown in Fig. 3(a). The velocity at S_0 takes the known velocity on the object, and the velocity at S_1 and S_2 are interpolated from four surrounding grid points using trilinear interpolation, as illustrated in Fig. 3(b). To keep S_1 and S_2 in different grid cells, $\Delta n > \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$, where Δx, Δy, and Δz are the grid spacings of the grid illustrated in Fig. 2(b).

It is shown in [38] that the normal force f_n satisfies

$$\frac{\partial f_n}{\partial \alpha_1} = -\left(\frac{2\vec{\Omega} - \vec{\omega}|_{S^+}}{ReJ^2}\right) \cdot \left(-\vec{T} \cdot \vec{B}\frac{\partial \vec{N}}{\partial \alpha_1} + J_1^2 \frac{\partial \vec{N}}{\partial \alpha_2}\right) - \frac{1}{ReJ}\left(\vec{T} \cdot \vec{B}\frac{\partial \omega_T}{\partial n}\bigg|_{S^+} - J_1^2 \frac{\partial \omega_B}{\partial n}\bigg|_{S^+}\right) + \vec{T} \cdot [\vec{q}], \tag{23}$$

$$\frac{\partial f_n}{\partial \alpha_2} = -\left(\frac{2\vec{\Omega} - \vec{\omega}|_{S^+}}{ReJ^2}\right) \cdot \left(\vec{T} \cdot \vec{B}\frac{\partial \vec{N}}{\partial \alpha_2} - J_2^2 \frac{\partial \vec{N}}{\partial \alpha_1}\right) - \frac{1}{ReJ}\left(-\vec{T} \cdot \vec{B}\frac{\partial \omega_B}{\partial n}\bigg|_{S^+} + J_2^2 \frac{\partial \omega_T}{\partial n}\bigg|_{S^+}\right) + \vec{B} \cdot [\vec{q}], \tag{24}$$

where $\vec{\omega} = (\omega_1, \omega_2, \omega_3) = \nabla \times \vec{v}$ is the vorticity, $\omega_T = \vec{\omega} \cdot \vec{T}$ and $\omega_B = \vec{\omega} \cdot \vec{B}$. We compute the surface vorticity $\vec{\omega}|_{S^+}$ above from

$$\begin{pmatrix} T_1 & T_2 & T_3 \\ B_1 & B_2 & B_3 \\ N_1 & N_2 & N_3 \end{pmatrix} \begin{pmatrix} \omega_1|_{S^+} \\ \omega_2|_{S^+} \\ \omega_3|_{S^+} \end{pmatrix} = \begin{pmatrix} \omega_T|_{S^+} \\ \omega_B|_{S^+} \\ \omega_N|_{S^+} \end{pmatrix}, \tag{25}$$

where $\omega_T|_{S^+}, \omega_B|_{S^+}$ and $\omega_N|_{S^+}$ are [38]

$$\omega_T|_{S^+} = -\frac{\partial \vec{v}}{\partial n}\bigg|_{S^+} \cdot (\vec{n} \times \vec{T}) + \vec{\Omega} \cdot \vec{T}, \tag{26}$$

$$\omega_B|_{S^+} = \frac{\partial \vec{v}}{\partial n}\bigg|_{S^+} \cdot (\vec{B} \times \vec{n}) + \vec{\Omega} \cdot \vec{B}, \tag{27}$$

$$\omega_N|_{S^+} = 2\vec{\Omega} \cdot \vec{N}. \tag{28}$$

Denote the right-hand sides of Eqs. (23) and (24) as r_1 and r_2. We can approximate $\frac{\partial \omega_T}{\partial n}|_{S^+}$ and $\frac{\partial \omega_B}{\partial n}|_{S^+}$ in r_1 and r_2 using one-sided finite differences similar to Eq. (22).

Eqs. (23) and (24) give the surface gradient (gradient along the boundary S) of f_n. The surface gradient operator $\nabla_\alpha = \left(\frac{\partial}{\partial \alpha_1}, \frac{\partial}{\partial \alpha_2}\right)$ can be inverted by solving the Poisson equation

$$\Delta_\alpha f_n = \frac{\partial r_1}{\partial \alpha_1} + \frac{\partial r_2}{\partial \alpha_2}, \tag{29}$$

where $\Delta_\alpha = \frac{\partial^2}{\partial \alpha_1^2} + \frac{\partial^2}{\partial \alpha_2^2}$ is a surface Laplace operator. This Poisson equation can be solved by FFT with periodic boundary conditions in both α_1 and α_2 directions [36].

Eqs. (23) and (24) indicate that computing f_n involves approximation of the normal derivatives of the vorticity. If the space resolution is insufficient to resolve the vorticity near the object, the computed pressure inside the object would not satisfy Eq. (14) due to the error in $[p] = f_n$. To improve the accuracy in the pressure, we can add a correction to f_n, as described in Appendix B. However, as demonstrated in Appendix B, the pressure outside the object seems less affected by this error, which makes the correction optional. Our results in Section 4 are obtained without the correction.

2.3. Fluid force and torque

To compute the fluid force $\vec{F}_f$ and torque $\vec{G}_f$ acting on the object by the flow, we use the following formulas:

$$\vec{F}_f = -\int_S \left(\vec{f}_{\tau b} + p|_{S^+}\vec{n}\right) J \mathrm{d}\alpha_1 \mathrm{d}\alpha_2, \tag{30}$$

$$\vec{G}_f = -\int_S (\vec{X} - \vec{x}_C) \times \left(\vec{f}_{\tau b} + p|_{S^+}\vec{n}\right) J \mathrm{d}\alpha_1 \mathrm{d}\alpha_2, \tag{31}$$

where the surface pressure $p|_{S^+}$ is interpolated according to Eq. (6) by

$$p|_{S^+}(S_0) = \frac{p(S_1) + p(S_{-1})}{2} + \frac{1}{2}\left([p] - \left[\frac{\partial p}{\partial n}\right]\Delta n\right) + O(\Delta n^2). \tag{32}$$

It is tempting to obtain $p|_{S^-}$ analytically from Eq. (14) and then replace $p|_{S^+}$ in Eqs. (30) and (31) by $p|_{S^+} = [p] + p|_{S^-} = f_n + p|_{S^-}$, which leads to the following formulas:

$$\vec{F}_f = -\int_S \vec{F} \mathrm{d}\alpha_1 \mathrm{d}\alpha_2 + V_o \frac{\mathrm{d}^2 \vec{x}_C}{\mathrm{d}t^2}, \tag{33}$$

$$\vec{G}_f = -\int_S (\vec{X} - \vec{x}_C) \times \vec{F} \mathrm{d}\alpha_1 \mathrm{d}\alpha_2, \tag{34}$$

where V_o is the volume of the object. They give the same results as Eqs. (30) and (31) if the correction to f_n is added. If the correction is not used, Eqs. (30) and (31) give better results for the reason that the pressure outside the object is less affected by the error in f_n, as demonstrated in Appendix B. In Section 4, we use Eqs. (30) and (31) to compute fluid force and torque.

3. Implementation

In this section, we briefly describe the implementation of our boundary condition capturing immersed interface method. More details on the implementation are referred to [36].

3.1. Jump conditions

The singular force and the discontinuous body force in Eqs. (10) and (11) induce the following principal jump conditions [34]

$$[\vec{v}] = 0, \quad \left[\frac{\partial \vec{v}}{\partial n}\right] = -Re\vec{f}_{\tau b}, \quad [\Delta v] = Re[\nabla p], \tag{35}$$

$$[p] = f_n, \quad \left[\frac{\partial p}{\partial n}\right] = \frac{1}{J}\left(\frac{\partial \tilde{f}_1}{\partial \alpha_1} + \frac{\partial \tilde{f}_2}{\partial \alpha_2}\right) + [\vec{q}] \cdot \vec{n}, \quad [\Delta p] = [s_p], \tag{36}$$

where $\tilde{f}_1$ and $\tilde{f}_2$ are defined previously in Eq. (18), and s_p is the right-hand side of Eq. (11). As shown in [34], the jump conditions of all the first and second Cartesian derivatives of $\vec{v}$ and p can be systematically derived from these principal jump conditions. Using Eqs. (4) and (5), we can then incorporate the Cartesian jump conditions in the finite difference discretization of all the first and second Cartesian derivatives in Eqs. (10) and (11) according to Eqs. (4) and (5).

To incorporate these Cartesian jump conditions, we need to find all the intersections of the boundary S and Cartesian grid lines. We use Lagrangian points to determine the intersections, where the Lagrangian points are introduced to track the motion of the boundary S.

We also use the Lagrangian points to compute geometric quantities for the boundary S, which are needed to derive the Cartesian jump conditions. Since the object is rigid, we only need to compute the geometric quantities once and relate the geometric quantities at two different time based on the transformation

$$\vec{X}(t) - \vec{x}_C(t) = R(t)(\vec{X}(0) - \vec{x}_C(0)), \tag{37}$$

where R is an orthogonal matrix describing the rotation of the object.

3.2. MAC scheme

We use the MAC scheme to solve the flow [9]. Specifically, we solve the momentum equation, Eq. (10), and the pressure Poisson equation, Eq. (11), on a uniform MAC grid illustrated in Fig. 2(b).

Define the central finite difference operators δ_x, and δ_{xx} on this grid as

$$\delta_x(\cdot)_{i,j,k} = \frac{(\cdot)_{i+\frac{1}{2},j,k} - (\cdot)_{i-\frac{1}{2},j,k}}{\Delta x} + c_x(\cdot)_{i,j,k}, \tag{38}$$

$$\delta_{xx}(\cdot)_{i,j,k} = \frac{(\cdot)_{i+1,j,k} - 2(\cdot)_{i,j,k} + (\cdot)_{i-1,j,k}}{\Delta x^2} + c_{xx}(\cdot)_{i,j,k}, \tag{39}$$

where c_x and c_{xx} are jump contributions determined according to Eqs. (4) and (5). If a finite difference stencil does not penetrate the boundary, the jump contributions are simply zero. The operators δ_y, δ_{yy}, δ_z, and δ_{zz} are defined similarly. The momentum equation for the velocity component u at $(i+\frac{1}{2},j,k)$ is discretized in space as

$$\frac{\partial u}{\partial t} = -\delta_x(uu) - \delta_y(vu) - \delta_z(wu) - \delta_x p + \frac{1}{Re}(\delta_{xx}+\delta_{yy}+\delta_{zz})u, \tag{40}$$

where the subscript $(i+\frac{1}{2},j,k)$ is omitted in the operators. The discretized equations for v at $(i,j+\frac{1}{2},k)$ and w at $(i,j,k+\frac{1}{2})$ are obtained similarly. The discretized pressure Poisson equation at (i,j,k) is

$$\begin{aligned}(\delta_{xx}+\delta_{yy}+\delta_{zz})p = &-\frac{\partial D}{\partial t} - 2\big(\delta_x(uD)+\delta_y(vD)+\delta_z(wD)\big) + \frac{1}{Re}(\delta_{xx}+\delta_{yy}+\delta_{zz})D \\ &+ 2\big(\delta_x u\delta_y v - \delta_y u\delta_x v + \delta_x u\delta_z w - \delta_z u\delta_x w + \delta_y v\delta_z w - \delta_z v\delta_y w\big).\end{aligned} \tag{41}$$

The MAC grid consists of one set of pressure grid points and three sets of velocity grid points. The staggered arrangement of the velocity and the pressure necessitates the interpolation of the velocity. Define the interpolation operators ε_i, ε_j, ε_k as

$$\varepsilon_i(\cdot)_{i,j,k} = \frac{(\cdot)_{i+\frac{1}{2},j,k} + (\cdot)_{i-\frac{1}{2},j,k}}{2} + c_i(\cdot)_{i,j,k}, \tag{42}$$

$$\varepsilon_j(\cdot)_{i,j,k} = \frac{(\cdot)_{i,j+\frac{1}{2},k} + (\cdot)_{i,j-\frac{1}{2},k}}{2} + c_j(\cdot)_{i,j,k}, \tag{43}$$

$$\varepsilon_k(\cdot)_{i,j,k} = \frac{(\cdot)_{i,j,k+\frac{1}{2}} + (\cdot)_{i,j,k-\frac{1}{2}}}{2} + c_k(\cdot)_{i,j,k}, \tag{44}$$

where c_i, c_j, and c_k are jump contributions calculated according to Eq. (6) when an interpolation stencil penetrates the boundary. We can then interpolate u in the following order:

$$u_{i,j,k} = \varepsilon_i u_{i,j,k}, \tag{45}$$

$$u_{i+\frac{1}{2},j+\frac{1}{2},k} = \varepsilon_j u_{i+\frac{1}{2},j+\frac{1}{2},k}, \tag{46}$$

$$u_{i+\frac{1}{2},j,k+\frac{1}{2}} = \varepsilon_k u_{i+\frac{1}{2},j,k+\frac{1}{2}}, \tag{47}$$

$$u_{i,j+\frac{1}{2},k} = \varepsilon_j u_{i,j+\frac{1}{2},k}, \tag{48}$$

$$u_{i,j,k+\frac{1}{2}} = \varepsilon_k u_{i,j,k+\frac{1}{2}}. \tag{49}$$

We can interpolate v and w similarly.

3.3. Numerical steps

It becomes clear at this point that the presence of the object (the boundary S) only affects the right-hand side of the linear system from the discretization since both the singular force and the body force are computed *explicitly*. We integrate the momentum equations in time using an explicit fourth-order RK scheme. We solve the discretized pressure Poisson equation using a direct FFT solver. A summary of the major components in each RK substep is given as follows with the computational cost in terms of the total MAC grid points M and the total Lagrangian points L.

- Calculate the tangential force $\vec{f}_{\tau b}$ using Eq. (21), $\mathcal{O}(L)$.
- Calculate the normal force f_n by solving Eq. (29) with FFT, $\mathcal{O}(L\ln L)$.
- Calculate the required Cartesian jump conditions for $\vec{v}$ and p on the Lagrangian points, $\mathcal{O}(L)$.
- Find all the intersections between the boundary S and MAC grid lines, $\mathcal{O}(L)$.
- Calculate jump contributions (the c's) to finite differences and interpolations, $\mathcal{O}(L)$.
- Solve the pressure p with FFT, $\mathcal{O}(M\ln M)$.
- Update the flow velocity $\vec{v}$, $\mathcal{O}(M)$.
- Update the position and velocity of the boundary S, $\mathcal{O}(L)$.

4. Results

Numerical examples are shown in this section to test the accuracy, efficiency and stability of our boundary condition capturing immersed interface method. In these examples, no correction to f_n is used, and Eqs. (30) and (31) are used to compute fluid force and torque.

4.1. Flow due to an oscillating sphere or torus

We consider the flow due to an oscillating sphere or torus in a box. The sphere or torus oscillates about the box center along the x-axis with the period T. The torus is oriented normal to the oscillation direction. The radius of the sphere is R. The

radius of the torus tube is 0.5R. The distance from the tube center to the torus center is 1.5R. The Reynolds number is defined as $Re = \frac{\rho U(2R)}{\mu}$, and the Stokes number $\epsilon = \sqrt{\frac{\pi\rho R^2}{T\mu}}$, where U is the amplitude of the oscillating velocity. The nondimensional oscillating period is $T_f = \frac{UT}{2R} = \frac{\pi Re}{4\epsilon^2}$. Both the sphere and the torus oscillate according to $x_C = -\frac{T_f}{2\pi}\cos\left(\frac{2\pi}{T_f}t\right)$.

4.1.1. Flow due to an oscillating sphere

We simulate the flow due to the oscillating sphere at $Re = 40$ and $\epsilon = 4$ with $T_f = \frac{\pi}{1.6}$. The computational domain has the sizes $16 \times 8 \times 8$ and is uniformly discretized with $256 \times 128 \times 128$ pressure grid points. The sphere is parametrized by 128×256 Lagrangian points. The time step of the simulation is $\Delta t = 0.01$.

Mei [20] simulated the flow at the same Re and ϵ in an unbounded domain using Fourier expansions. After the flow becomes periodic, we compare the drag on the sphere in our simulation with that by Mei [20]. Fig. 4 indicates a very good agreement.

On an Intel Pentium D 3.20GHz dual core processor, it takes about 0.9 h of CPU time (0.45 wall-clock time) to simulate the flow for one time unit on the $256 \times 128 \times 128$ grid. As a comparison, the immersed boundary method by Mittal et al. [22] requires about 7 h of CPU time to simulate the flow past a sphere on a single processor of 64-bit 1.8GHz AMD Opteron computer for one time unit (based on the sphere diameter and the incoming flow speed) on a $192 \times 120 \times 120$ grid.

4.1.2. Flow due to an oscillating torus

We simulate the flow due to the oscillating torus at $Re = 20$ with $T_f = 2$ for convergence analysis. The domain has the sizes $4 \times 4 \times 4$ and is discretized with $n \times n \times n$ pressure grid points. The torus is parametrized by $m \times (2\text{ m})$ Lagrangian points. The resolution of a simulation is represented by (n,m). We simulate the flow from $t = 0$–2 at the resolutions $R1 = (33,32)$, $R2 = (66,64)$, and $R3 = (132,128)$. The corresponding time steps are $\Delta t = 0.001$, 0.0005, and 0.00025.

Let $q^{(R1)}$, $q^{(R2)}$ and $q^{(R3)}$, where $q = u$, v, w or p, be the discrete solutions at the resolutions $R1$, $R2$ and $R3$, respectively. We define as follows the L_∞ and L_2 norm of the difference of two solutions at the same locations

$$\|E^{(R1-R2)}\|_\infty = \max_{(i,j,k)\in\mathcal{O}_T}\left|q^{(R1)}_{i,j,k} - q^{(R2)}_{2i-1,2j-1,2k-1}\right|, \tag{50}$$

$$\|E^{(R2-R3)}\|_\infty = \max_{(i,j,k)\in\mathcal{O}_T}\left|q^{(R2)}_{2i-1,2j-1,2k-1} - q^{(R3)}_{4i-3,4j-3,4k-3}\right|, \tag{51}$$

$$\|E^{(R1-R2)}\|_2^2 = \frac{1}{\mathcal{N}}\sum_{(i,j,k)\in\mathcal{O}_T}\left|q^{(R1)}_{i,j,k} - q^{(R2)}_{2i-1,2j-1,2k-1}\right|^2, \tag{52}$$

$$\|E^{(R2-R3)}\|_2^2 = \frac{1}{\mathcal{N}}\sum_{(i,j,k)\in\mathcal{O}_T}\left|q^{(R2)}_{2i-1,2j-1,2k-1} - q^{(R3)}_{4i-3,4j-3,4k-3}\right|^2, \tag{53}$$

where $i,j,k \in \{1,\ldots,33\}$, $\mathcal{O}_T$ is the set of the points (i,j,k) outside the torus at the resolution $R1 = (33,32)$, and $\mathcal{N}$ is the total number of such points. The velocity at the pressure grid points is obtained by interpolation. The pressure p is solved subject to a constant. To make the norms for the pressure meaningful, we subtract the pressure by its average outside the torus. Since a time step is much smaller than a space step in each simulation, the error from spatial discretization dominates, and we can estimate the order of the spatial accuracy as

$$\text{order} = \log_2\frac{\|E^{(R1-R2)}\|}{\|E^{(R2-R3)}\|}, \tag{54}$$

where $\|\cdot\|$ is either the L_∞ or L_2 norm.

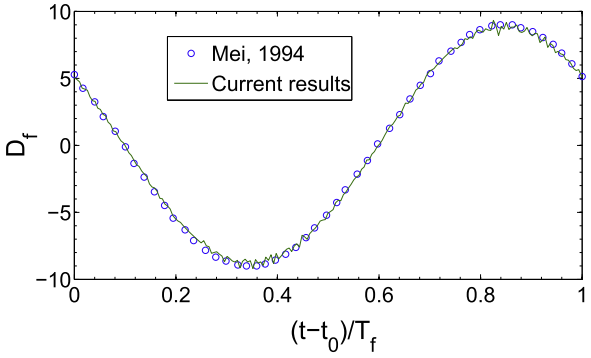

Fig. 4. Comparison of the periodic drag (normalized by the Stokes drag) on an oscillating sphere.

Table 1
Spatial convergence analysis for the flow due to an oscillating torus.

	u	Order	v (or w)	Order	p	Order
$\|E^{(R1-R2)}\|_\infty$	6.18×10^{-2}		8.03×10^{-2}		2.48×10^{-1}	
$\|E^{(R2-R3)}\|_\infty$	1.74×10^{-2}	1.83	2.36×10^{-2}	1.77	9.66×10^{-2}	1.36
$\|E^{(R1-R2)}\|_2$	6.57×10^{-3}		3.22×10^{-3}		7.03×10^{-2}	
$\|E^{(R2-R3)}\|_2$	2.07×10^{-3}	1.67	8.91×10^{-4}	1.85	2.77×10^{-2}	1.34

The results in Table 1 indicate that the accuracy of the velocity is almost second order in both norms, and the accuracy of the pressure is above first order. The comparison of these results with Table 4 in Appendix B suggests that the correction to f_n can increase the accuracy of the pressure to near second order but has little effect on the velocity.

4.2. Flow due to a rotating sphere

In this example, we simulate the steady flow driven by a rotating sphere at the center of a box of the sizes $20 \times 20 \times 20$. The sphere has the radius R. It rotates with the constant angular velocity Ω_x about a diameter on the x-axis. The Reynolds number of the flow is defined as $Re = \frac{\rho(\Omega_x R)R}{\mu}$. We carry out simulations at Re = 20, 50, and 100. We represent the sphere using 128×256 Lagrangian points. We use $128 \times 128 \times 128$ pressure grid points for Re = 20 and 50 and $256 \times 256 \times 256$ for Re = 100. The time step of the simulations is Δt = 0.02.

The main feature of the flow is an inflow toward the poles on the x-axis balanced by an outflow near the equator on the yz-plane, as shown in Fig. 5. The extent of the polar inflow is reflected by the critical angle θ_c, which is measured from the rotation x-axis and the radius through the center of a toroidal vertex ring, as sketched in Fig. 5.

Dennis et al. [3] used polar coordinates and series expansions to solve the steady axisymmetric flow in an unbounded domain. Wang et al. [33] solved the flow in 3D using a hybrid meshfree-and-Cartesian grid method. When the flow in our simulations becomes steady, the critical angle θ_c and the torque coefficient $C_G = \frac{-\overline{G}_{fx}}{0.5\rho R^5 \Omega_x^2}$, where $\overline{G}_{fx}$ is the x component of the dimensional fluid torque on the sphere, agree with those by Dennis et al. [3] and Wang et al. [33], as shown in Table 2. In Fig. 6, we compare our radial velocity profiles in the equatorial plane with those by Dennis et al. [3]. The agreement is also very good.

4.3. Flow past a torus

We simulate the flow past a stationary torus. The torus is oriented normal to the inflow in the x direction. The Reynolds number is defined as $Re = \frac{\rho U(2R)}{\mu}$, where U is the inflow speed, and R is the radius of the torus tube. The aspect ratio A_R of the torus is defined as the ratio of the mean radius of the torus to the radius of the tube.

The computational domain has the sizes $24 \times 12 \times 12$. The aspect ratio of the torus is A_R = 2. The center of the torus is the origin of the Cartesian coordinate system, and it is away from the inlet and the four sides of the domain by equal distance. We use $512 \times 256 \times 256$ pressure grid points and 128×256 Lagrangian points. The time step of the simulation is controlled by the CFL numbers $\mathrm{CFL}_c = \mathrm{CFL}_d = 0.5$, where $\mathrm{CFL}_c = \Delta t\left(\frac{u_{\max}}{\Delta x} + \frac{v_{\max}}{\Delta y} + \frac{w_{\max}}{\Delta z}\right)$ and $\mathrm{CFL}_d = \frac{\Delta t}{Re}\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right)$. The Neumann condi-

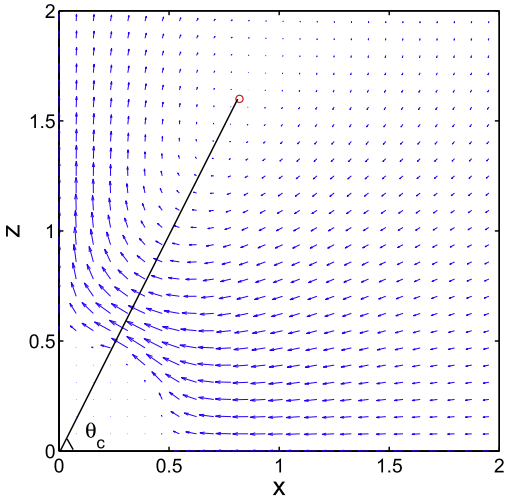

Fig. 5. Steady velocity field at the central xz plane of the flow around a rotating sphere.

Table 2
Comparisons of the critical angle θ_c and the torque coefficient C_G.

	$Re = 20$		$Re = 50$		$Re = 100$	
	θ_c	C_G	θ_c	C_G	θ_c	C_G
Our results	62.9	2.984	69.4	1.544	73.7	1.014
Dennis et al. [3]	62.6	3.048	69.4	1.544	73.8	0.966
Wang et al. [33]	62.1	2.990	69.7	1.535	74.2	0.956

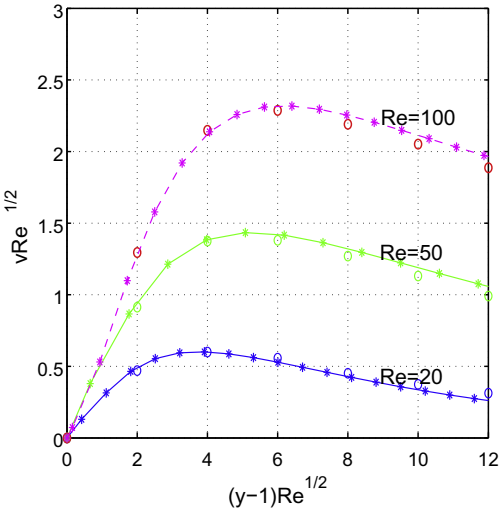

Fig. 6. Variation of the radial velocity with the radial distance in the equatorial plane. Lines with stars: our results, open circles: Dennis et al. [3].

tion $\frac{\partial p}{\partial x} = \frac{1}{Re}\frac{\partial^2 u}{\partial x^2}$ is applied at the inlet. Symmetry conditions are applied at the four sides. At the outlet, the conditions $\frac{\partial \vec{v}}{\partial x} = 0$ and $\frac{\partial p}{\partial x} = \frac{1}{Re}\frac{\partial^2 u}{\partial x^2}$ are used.

Fig. 7 shows the time history of the fluid force on the torus at $Re = 10$, 20, 100, and 200. The drag coefficient C_D is defined as $C_D = \frac{2F_{fx}}{A_f}$, where $A_f = \pi A_R$ is the projected frontal area of the torus. The side force coefficient C_S is defined similarly. The drag coefficient becomes a constant at $Re = 10$, 20, and 100 after the flow becomes steady. Its values agree with those by Sheard et al. [28], as shown in Table 3. At $Re = 200$, the flow becomes unsteady with time-varying C_D and C_S as indicated in Fig. 7. In Fig. 8, we plot the contours of vorticity components in two central planes. Asymmetry is observed in both planes. The asymmetry is more prominent at the xy-plane than the xz-plane, which is consistent with fact that the y-component of the fluid force is larger than the z-component (not shown here).

4.4. Flow around a flapper

In the last example, we simulate the flow around a 3D flapper to compare it with similar 2D simulations and to test the stability of our method at high Reynolds numbers. The flapper is a rounded plate with aspect ratio 6:1:0.25. We chose the

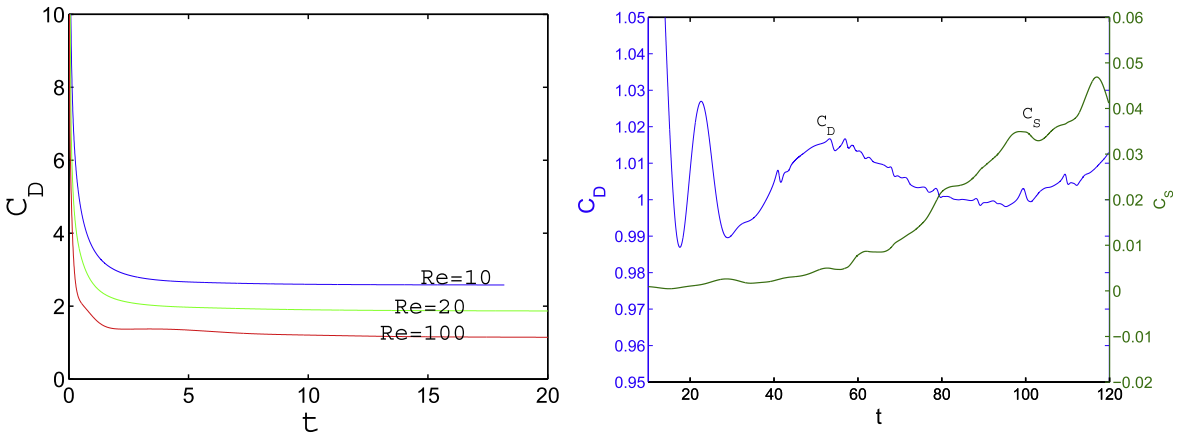

Fig. 7. The fluid force on the torus in an incoming flow.

Table 3
Drag on a torus in an incoming flow.

	Re	10	20	100
C_D	Our results	2.58	1.86	1.11
	Sheard et al. [28]	2.5	1.8	0.9

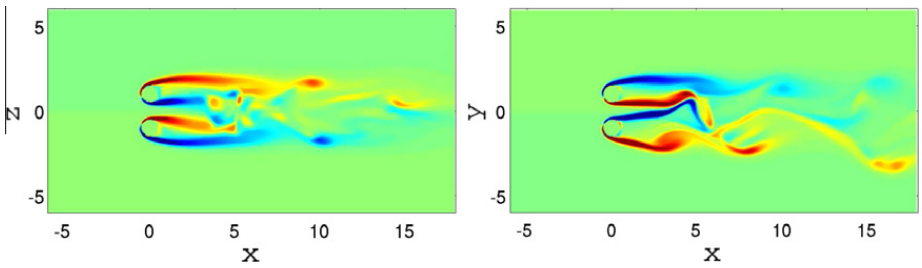

Fig. 8. Contours of vorticity components in the flow past a torus at $Re = 200$. Left: the y-component at the xz-plane. Right: the z-component at the xy-plane.

width W of the flapper as the length scale and the maximum speed U at the flapper center as the velocity scale to define the Reynold number $Re = \frac{\rho UW}{\mu}$. The flapper undergoes a 2D sinusoidal flapping motion in its transverse plane with the nondimensional flapping period $T_f = 2.5\pi$. The flapping motion is given by

$$x_C = 0, \tag{55}$$

$$y_C = 1.25(\cos(0.8t) + 1)\cos\left(\frac{\pi}{3}\right), \tag{56}$$

$$z_C = 1.25(\cos(0.8t) + 1)\sin\left(\frac{\pi}{3}\right), \tag{57}$$

$$\theta = \frac{3\pi}{4} + \frac{\pi}{4}\sin(0.8t), \tag{58}$$

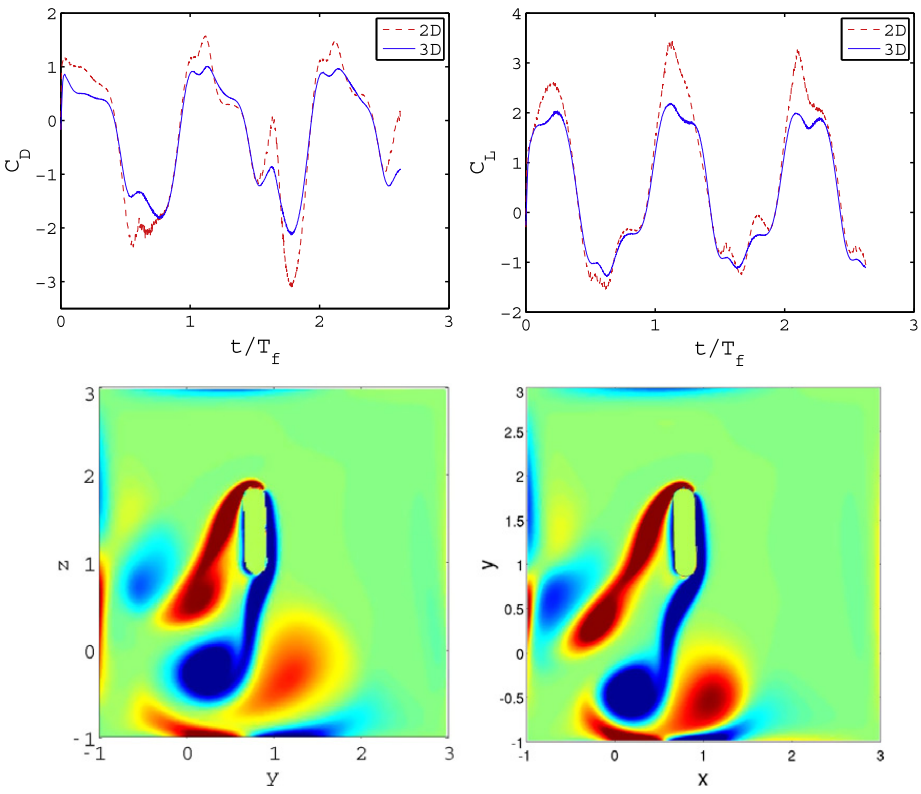

Fig. 9. Comparisons between 2D and 3D simulations of the flow around a flapper. Top: The drag and lift. Bottom: The spanwise vorticity at the central transverse plane in the 3D simulation (left) and the vorticity in the 2D simulation (right) after 0.8 flapping period from the start.

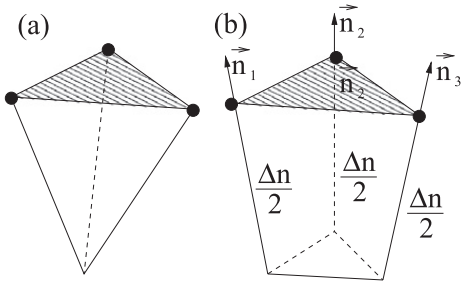

Fig. 10. Geometries used in the point-in-object test: (a) a pyramid formed by a reference point and a triangular patch, and (b) a wedge formed from a triangular patch.

Table 4
Spatial convergence analysis for the flow due to an oscillating torus simulated with the correction.

	u	Order	$v(w)$	Order	p	Order
$\|\|E^{(R1-R2)}\|\|_\infty$	6.23×10^{-2}		7.75×10^{-2}		1.45×10^{-1}	
$\|\|E^{(R2-R3)}\|\|_\infty$	1.70×10^{-2}	1.87	2.31×10^{-2}	1.75	4.50×10^{-2}	1.69
$\|\|E^{(R1-R2)}\|\|_2$	6.43×10^{-3}		3.10×10^{-3}		4.10×10^{-2}	
$\|\|E^{(R2-R3)}\|\|_2$	2.02×10^{-3}	1.67	8.54×10^{-4}	1.86	1.17×10^{-2}	1.81

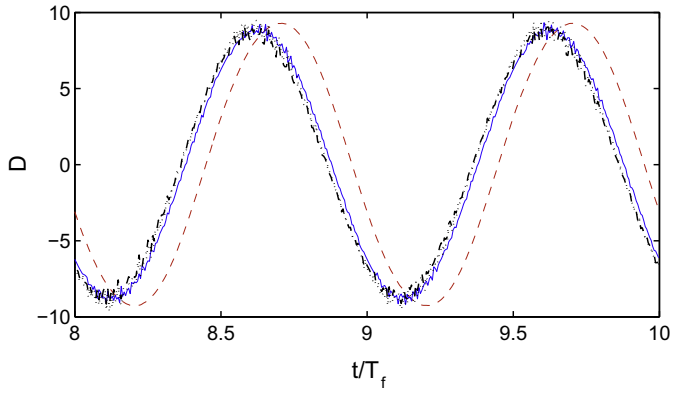

Fig. 11. Time history of the drag on a sphere oscillating with the period T_f. dashdot line: Eq. (30) with the correction, dotted line: Eq. (33) with the correction; solid line: Eq. (30) without the correction, dashed line: Eq. (33) without the correction.

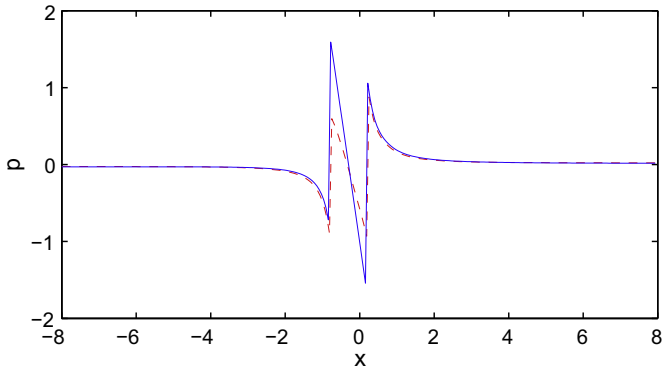

Fig. 12. Pressure along the x-axis after 10 oscillating periods in a flow due to an oscillating sphere. Solid line: with the correction, dashed line: without the correction.

where θ is the pitching angle measured from the xy-plane. The flapper flaps in a box with the sizes $8 \times 4 \times 4$. The box is discretized with $240 \times 240 \times 240$ pressure grid points. The flapper is represented by 64×128 Lagrangian points. The time step is controlled by $CFL_c = CFL_d = 0.5$.

The flow around this flapper at $Re = 157$ is compared with the flow around a corresponding 2D flapper [32,35,37] in Fig. 9. The 2D flapper undergoes the same sinusoidal flapping motion as the 3D flapper, and has the same shape as the transverse cross section of the 3D flapper. The 2D simulation has been well validated and documented [32,35,37]. The difference between the 3D and the 2D simulations is expected, but the overall agreement is good.

To test the stability of our method, we change the Reynolds number to $Re = 1570$ without changing the spatial resolution and the CFL numbers. The simulation at this much higher Reynolds number remains stable, even though the flow is not resolved with the low resolution.

5. Conclusions

We propose a boundary condition capturing immersed interface method to simulate a flow around moving rigid objects. The effect of a rigid object is represented as a singular force. We determine the singular force *explicitly*. We compute its tangential component from the normal derivative of the flow velocity, and we solve its normal component from a 2D Poisson equation on the surface of the object. We also employ a discontinuous body force to attain the rigid motion of the fluid replacing the object. It is *simple* and *straighforward* to implement our method in an existing CFD code of the immersed interface method.

We test the *accuracy*, *efficiency*, and *stability* of our method in some canonical examples. The test results indicate that (1) our method predicts flow behaviors correctly, (2) its efficiency for a flow around moving objects is almost the same as for a simple cavity flow, (3) its accuracy based on the infinity norm is almost second order for the velocity and above first order for the pressure, and (4) our method is stable at high Reynolds numbers under the standard CFL condition. We can also add the correction to the singular force to improve the pressure accuracy to almost second order.

We attribute the boundary condition capturing feature to the idea proposed in [37]. The method in this paper is a significant extension of this idea from 2D to 3D. It utilizes the necessary new formulas derived in [38] and consists of new ways to implement these formulas. In particular, the new formulas only need explicit approximations of the normal derivatives of the velocity and the vorticity; the normal component of the singular force is determined by inverting a surface gradient operator with FFT; and the *optional* correction to the normal singular force is solved by GMRES. In addition, we suggest more accurate formulas than in [37] to compute fluid force and torque for simulations without the correction.

Acknowledgments

This work is supported by the NSF Grant DMS 0915237.

Appendix A. Point-in-object test

Consider N_B rigid objects, marked as $1,2,\ldots,$ and N_B. We can test whether a grid point falls inside them using the fact that the coordinates on their boundaries satisfy the rigid transformation given by Eq. (37).

We first test if a grid point is inside the object m ($m = 1,2,\ldots,N_B$) at the reference time *0* by using each pyramid of the object shown in Fig. 10(a). A pyramid is formed by a reference point chosen for the object and a triangular patch on the boundary of the object. We record the test outcome as an integer Φ. If the grid point is inside or at the boundary, then $\Phi = m$. If it is outside all the N_B objects, then $\Phi = 0$. To assign values to Φ, we initially set $\Phi = 0$ for every grid point. We then consider a grid point in the box enveloping a pyramid of the object m and set $\Phi = m - \Phi$ for the grid point if it is inside the pyramid. If this grid point is inside pyramids for odd-number times, it is inside the object m ($\Phi = m$). Otherwise it is outside. This test is valid for both simply and multiply connected objects. We make sure that a point on a face shared by two pyramids is not tested twice.

We now consider a grid point $\vec{x}$ in the box enveloping the object m at a later time t. To test if the grid point is inside at the time t, we transform its coordinates $\vec{x}$ to $\vec{x}_0$ as

$$\vec{x}_0 - \vec{x}_C(0) = R(t)^{-1}(\vec{x} - \vec{x}_C(t)). \tag{59}$$

We then determine in which gird cell the point $\vec{x}_0$ falls. If $\Phi = m$ ($\Phi = 0$) at all eight vertices of the cell, the point $\vec{x}_0$ is inside the object m (outside all the N_B objects) at the time *0*, and the grid point $\vec{x}$ is inside the object m (outside all the N_B objects) at the time t. If $\Phi = m$ at some vertices and $\Phi = 0$ at the others of the cell, the cell is cut by the boundary. In this latter case, we regard the grid point $\vec{x}$ outside all the N_B objects at the time t, and then perform an extra test to see if it is inside an object.

In the extra test, we consider the object m at the time t directly. We form a wedge using each triangular patch and the normals at the three vertices of the patch, as shown in Fig. 10(b). We choose Δn to guarantee a grid point in a cell cut by the boundary is contained in a wedge. We then test if a grid point in the box enveloping the wedge is in this wedge. It it is, it is also inside the object m.

Appendix B. Correction for f_n

For a 2D flow with a 2D moving object, the correction δf_n for f_n can be found by solving a Neumann-to-Dirichlet map using the boundary element method [37] at every time step. It is expensive to do so in 3D, so we use an augmented variable approach [19] in 3D. We set δf_n as an augmented variable and solve δf_n using GMRES.

We split the pressure p into a Poisson part p_P and a Laplace part p_L as $p = p_P + p_L$. We first solve the Poisson part p_P from

$$\Delta p_P = \Delta p, \tag{60}$$

$$\left[\frac{\partial p_P}{\partial n}\right] = \left[\frac{\partial p}{\partial n}\right], \tag{61}$$

$$[p_P] = f_n, \tag{62}$$

where f_n is obtained from the predictor given by Eq. (29). The Laplace part p_L satisfies

$$\Delta p_L = 0, \tag{63}$$

$$\left[\frac{\partial p_L}{\partial n}\right] = 0, \tag{64}$$

$$[p_L] = \delta f_n, \tag{65}$$

where δf_n is the correction. The correction δf_n is found by requiring

$$p_L|_{S^-} = p|_{S^-} - p_P|_{S^-}, \tag{67}$$

where $p|_{S^-}$ is obtained from Eq. (14) analytically, and $p_P|_{S^-}$ is interpolated from p_P. Using the stencil in Fig. 3(a), the interpolation scheme is given by

$$p_P(S_0)|_{S^-} = \frac{p_P(S_1) + p_P(S_{-1})}{2} - \frac{1}{2}\left([p_P] + \left[\frac{\partial p_P}{\partial n}\right]\Delta n\right) + O(\Delta n^2). \tag{68}$$

Let $\underline{\delta f_n}$ denote the vector formed by the values of δf_n at all the Lagrangian points. Let $\underline{\underline{p_L}}$ denote the vector formed by the values of p_L at the MAC pressure grid points. We have

$$\Delta_h \underline{\underline{p_L}} = C\underline{\delta f_n}, \tag{69}$$

where Δ_h is the discrete Laplace operator, and the matrix–vector product $C\underline{\delta f_n}$ is the jump contribution to this operator. The interpolation of $p_L|_{S^-}$ using a scheme similar to Eq. (68) can be written as $E\underline{\underline{p_L}}$. By Eq. (67), we then have the linear system

$$E\Delta_h^{-1} C\underline{\delta f_n} = \underline{p|_{S^-}} - \underline{p_P|_{S^-}}, \tag{70}$$

from which we can solve $\underline{\delta f_n}$. This linear system is well-conditioned and can be solved by GMRES. However, if extrapolation (instead of the interpolation) is used to obtain $p_L|_{S^-}$, our numerical experiments indicate that a simulation can become unstable.

Using the correction, we can achieve almost second-order accuracy in both the velocity and the pressure, as shown in Table 4. Fig. 11 shows the unsteady drag on an oscillating sphere in two simulations, one with the correction and the other without. In each simulation, we calculate the drag using two formulas, one by Eq. (30) and the other by Eq. (33). The two formulas give almost the same results when the correction is turned on. Without the correction, there appears a small phase shift in the drag when Eq. (33) is used, which is substantially reduced if Eq. (30) is used instead. The pressure outside the sphere is much less sensitive to the correction than the pressure inside, as indicated in Fig. 12, which explains why Eq. (30) gives good drag prediction for the simulation without the correction.

References

[1] R.P. Beyer, R.J. Leveque, Analysis of a one-dimensional model for the immersed boundary method, SIAM J. Numer. Anal. 29 (2) (1992) 332–364.
[2] Jung-Il Choi, Roshan C. Oberoia, Jack R. Edwardsa, Jacky A. Rosatib, An immersed boundary method for complex incompressible flows, J. Comput. Phys. 224 (2007) 757–784.
[3] S.C.R. Dennis, S.N. Singh, D.B. Ingham, The steady flow due to a rotating sphere at low and moderate Reynolds numbers, J. Fluid Mech. 101 (1980) 257–279.
[4] 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.
[5] Anvar Gilmanov, Fotis Sotiropoulos, A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies, J. Comput. Phys. 207 (2005) 457–492.
[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] Boyce E. Griffith, Charles S. Peskin, On the order of accuracy of the immersed boundary method: Higher order convergence rates for sufficiently smooth problems, J. Comput. Phys. 208 (2005) 75–105.
[8] Robert D. Guy, David A. Hartenstine, On the accuracy of direct forcing immersed boundary methods with projection methods, J. Comput. Phys. 229 (2010) 2479–2496.
[9] Francis H. Harlow, J. Eddie Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Phys. Fluids 8 (12) (1965) 2182–2189.

[10] Thomas Y. Hou, Zuoqiang Shi, An efficient semi-implicit immersed boundary method for the Navier–Stokes equations, J. Comput. Phys. 227 (2008) 8968–8991.
[11] K. Ito, M.-C. Lai, Z. Li, A well-conditioned augmented system for solving Navier–Stokes equations in irregular domains, J. Comput. Phys. 228 (2009) 2616–2628.
[12] Jungwoo Kim, Dongjoo Kim, Haecheon Choi, An immersed-boundary finite-volume method for simulations of flow in complex geometries, J. Comput. Phys. 171 (2001) 132–150.
[13] Ming-Chih Lai, Charles S. Peskin, An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, J. Comput. Phys. 160 (2000) 705–719.
[14] 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.
[15] Randall J. LeVeque, Zhilin Li, The immersed interface method for elliptic equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 31 (4) (1994) 1019–1044.
[16] Randall J. LeVeque, Zhilin Li, Immersed interface methods for Stokes flow with elastic boundaries or surface tension, SIAM J. Sci. Comput. 18 (3) (1997) 709–735.
[17] Zhilin Li, Ming-Chih Lai, The immersed interface method for the Navier–Stokes equations with singular forces, J. Comput. Phys. 171 (2001) 822–842.
[18] Zhilin Li, Kazufumi Ito, The immersed interface method – numerical solutions of PDEs involving interfaces and irregular domains, SIAM Front. Appl. Math. 33 (2006). ISBN: 0-89971-609-8.
[19] Z. Li, X. Wan, K. Ito, S.R. Lubkin, An augmented approach for the pressure boundary condition in a Stokes flow, Commun. Comput. Phys. 1 (1) (2006) 874–885.
[20] Renwei Mei, Flow due to an oscillating sphere and an expression for unsteady drag on the sphere at finite Reynolds number, J. Fluid Mech. 270 (1994) 133–174.
[21] Rajat Mittal, Gianluca Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (2005) 239–261.
[22] R. Mittal, H. Dong, M. Bozkurttas, F.M. Najjar, A. Vargas, A. von Loebbecke, A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries, J. Comput. Phys. 227 (2008) 4825–4852.
[23] E.P. Newren, A.L. Fogelson, R.D. Guy, R.M. Kirby, Unconditionally stable discretizations of the immersed boundary equations, J. Comput. Phys. 222 (2007) 702–719.
[24] Charles S. Peskin, Flow patterns around heart valves: a numerical method, J. Comput. Phys. 10 (1972) 252–271.
[25] Charles S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220–252.
[26] Charles S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.
[27] E.M. Saiki, S. Biringen, Numerical simulation of a cylinder in uniform flow: application of a virtual boundary method, J. Comput. Phys. 123 (1996) 450–465.
[28] G.J. Sheard, K. Hourigan, M.C. Thompson, Computations of the drag coefficients for low-Reynolds-number flow past rings, J. Fluid Mech. 526 (2005) 257–275.
[29] John M. Stockie, Brian R. Wetton, Analysis of stiffness in the immersed boundary method and implications for time-stepping schemes, J. Comput. Phys. 154 (1999) 41–64.
[30] Yu-Heng Tseng, Joel H. Ferziger, A ghost-cell immersed boundary method for flow in complex geometry, J. Comput. Phys. 192 (2003) 593–623.
[31] H.S. Udaykumar, R. Mittal, P. Rampunggoon, A. Khanna, A sharp interface Cartesian grid method for simulating flows with complex moving boundaries, J. Comput. Phys. 174 (2001) 345–380.
[32] Z. Jane Wang, Two dimensional mechanism for insect hovering, Phys. Rev. Lett. 85 (10) (2000) 2216–2219.
[33] X.Y. Wang, P. Yu, K.S. Yeo, B.C. Khoo, SVD-GFD scheme to simulate complex moving body problems in 3D space, J. Comput. Phys. 229 (2010) 2314–2338.
[34] Sheng Xu, Z. Jane Wang, Systematic derivation of jump conditions for the immersed interface method in three-dimensional flow simulation, SIAM J. Sci. Comput. 27 (6) (2006) 1948–1980.
[35] Sheng Xu, Z. Jane Wang, An immersed interface method for simulating the interaction of a fluid with moving boundaries, J. Comput. Phys. 216 (2) (2006) 454–493.
[36] Sheng Xu, Z. Jane Wang, A 3D immersed interface method for fluid–solid interaction, Comput. Methods Appl. Mech. Eng. 197 (2008) 2068–2086.
[37] Sheng Xu, The immersed interface method for simulating prescribed motion of rigid objects in an incompressible viscous flow, J. Comput. Phys. 227 (2008) 5045–5071.
[38] Sheng Xu, Singular forces in the immersed interface method for rigid objects in 3D, Appl. Math. Lett. 22 (2009) 827–833.
[39] Jianming Yang, Elias Balaras, An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries, J. Comput. Phys. 215 (2006) 12–40.
[40] T. Ye, R. Mittal, H.S. Udaykumar, W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries, J. Comput. Phys. 156 (1999) 209–240.
[41] N. Zhang, Z.C. Zheng, An improved direct-forcing immersed-boundary method for finite difference applications, J. Comput. Phys. 221 (2007) 250–268.