Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. xxx (2007) xxx–xxx
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

A 3D immersed interface method for fluid–solid interaction

Sheng Xu [a,*], Z. Jane Wang [b]

[a] Department of Mathematics, Southern Methodist University, P.O. 750156, Dallas, TX 75275-0156, United States
[b] Department of Theoretical and Applied Mechanics, Cornell University, Ithaca, NY 14853, United States

Received 20 February 2007; received in revised form 4 June 2007; accepted 17 June 2007

This paper is written in honor of Professor Charles Peskin's 60th birthday.

Abstract

In immersed interface methods, solids in a fluid are represented by singular forces in the Navier–Stokes equations, and flow jump conditions induced by the singular forces directly enter into numerical schemes. This paper focuses on the implementation of an immersed interface method for simulating fluid–solid interaction in 3D. The method employs the MAC scheme for the spatial discretization, the RK4 scheme for the time integration, and an FFT-based Poisson solver for the pressure Poisson equation. A fluid–solid interface is tracked by Lagrangian markers. Intersections of the interface with MAC grid lines identify finite difference stencils on which jump contributions to finite difference schemes are needed. To find the intersections and to interpolate jump conditions from the Lagrangian markers to the intersections, parametric triangulation of the interface is used. The velocity of the Lagrangian markers is interpolated directly from surrounding MAC grid nodes with interpolation schemes accounting for jump conditions. Numerical examples demonstrate that (1) the method has near second-order accuracy in the infinity norm for velocity, and the accuracy for pressure is between first and second order; (2) the method conserves the volume enclosed by a no-penetration boundary; and (3) the method can efficiently handle multiple moving solids with ease.
Published by Elsevier B.V.

Keywords: Immersed interface method; Immersed boundary method; Fluid–solid interaction; Singular forces jump conditions

1. Introduction

Immersed interface methods are offspring of the immersed boundary method. The original immersed boundary method was proposed by Peskin to simulate blood flow in the human heart [20,21]. It treats heart walls and heart valves as fiber-reinforced fluid. The immersed boundary method is therefore a mathematical formulation, in which the effects of solid boundaries are formulated as forces in the Navier–Stokes equations. The forces are determined from boundary configurations according to constitutive laws. They involve the form of the Dirac δ function and are thus called singular forces. The immersed boundary method has been applied to a wide variety of problems, especially biological flows, as summarized in [23]. When applied to flow simulation, an immersed interface method shares the same mathematical formulation as the immersed boundary method, which reads

$$\frac{\partial \mathbf{v}}{\partial t} + \nabla \cdot (\mathbf{v}\mathbf{v}) = -\nabla p + \frac{1}{Re}\Delta \mathbf{v} + \int_B \mathbf{f}(\alpha_1, \alpha_2, t)\delta(\mathbf{x} - \mathbf{X}(\alpha_1, \alpha_2, t))\, d\alpha_1\, d\alpha_2, \tag{1}$$

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

where $\mathbf{v}$ is velocity, p is pressure, t is time, Re is the Reynolds number, B is the boundary of a solid, $\delta(\mathbf{x} - \mathbf{X}(\alpha_1, \alpha_2, t))$ is the 3D Dirac δ function, $\mathbf{x}$ is Cartesian coordinates, $\mathbf{X}(\alpha_1, \alpha_2, t)$ is the coordinates of the boundary,

* Corresponding author. Tel.: +1 214 768 2985; fax: +1 214 768 2355.
E-mail addresses: sxu@smu.edu (S. Xu), zw24@cornell.edu (Z. Jane Wang).

0045-7825/$ - see front matter Published by Elsevier B.V.
doi:10.1016/j.cma.2007.06.012

and **f** is the density of a singular force in the parameter space which is formed by two Lagrangian parameters α_1 and α_2 parameterizing the boundary B as shown in Fig. 1. In the above formulation, only one boundary, the boundary B, is considered. This formulation is used hereafter for the presentation of this paper. If multiple boundaries are considered, they can be easily included in the same manner.

In the immersed boundary method, the boundary of an immersed solid is tracked by Lagrangian markers that are convected by a fluid. Numerically, the communication between the solid and the fluid is obtained by spreading the singular forces from the Lagrangian markers to nearby Cartesian grid nodes and interpolating the velocity from nearby Cartesian grid nodes to the Lagrangian markers with the use of discrete Dirac δ functions. Many research efforts have been devoted to analyze and improve the accuracy, stability, conservation, and robustness of the immersed boundary method [3,28,22,25,27,5,12,34]. Motivated to improve the accuracy of the immersed boundary method from first order to second order, LeVeque and Li [15,16] proposed immersed interface methods. To avoid the use of discrete Dirac δ functions, an immersed interface method directly incorporates singularity-induced jump conditions of flow quantities into finite difference schemes, which gives it second-order or higher accuracy, sharp fluid–solid interfaces, and very good conservation of mass enclosed by no-penetration boundaries.

Immersed interface methods were initially proposed for elliptic equations [15] and the Stokes equations [16]. Later, they were extended to 1D nonlinear parabolic equations [30], Poisson equations with Neumann boundary conditions [8], elliptic equations with variable coefficients [31,6,2], and the 2D Navier–Stokes equations [17,14,19,33,13]. These various methods are summarized in the recent book by Li and Ito [18].

To extend immersed interface methods to the 3D Navier–Stokes equations, necessary jump conditions have been systematically derived by Xu and Wang [32]. A list of these jump conditions is given in Section 3. The incorporation of jump conditions into finite difference schemes is based on the following generalized Taylor expansion [32]:

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

where $g(s)$ is a non-smooth and discontinuous function as shown in Fig. 2a, and $[g^{(n)}(s_l)]$ denotes jump conditions along the s-axis, i.e. $[g^{(n)}(s_l)] = g^{(n)}(s^+_l) - g^{(n)}(s^-_l)$. Second-order central finite difference schemes with discontinuities at ξ and η on its stencil shown in Fig. 2b can be modified as follows to keep their second-order accuracy:

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

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

An interpolation scheme also needs to account for jump conditions if its interpolation stencil contains discontinuities. The following second-order interpolation scheme applies to the case shown in Fig. 2b

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

The jump conditions derived in [32] have been employed in an immersed interface method to simulate the interaction

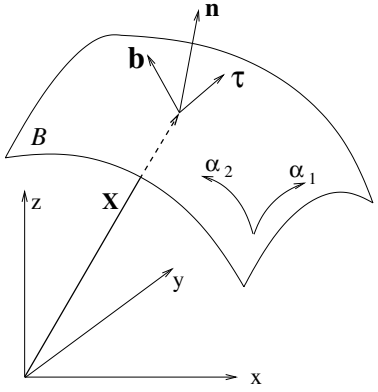


Fig. 1. A parametrized boundary in a Cartesian coordinate system.

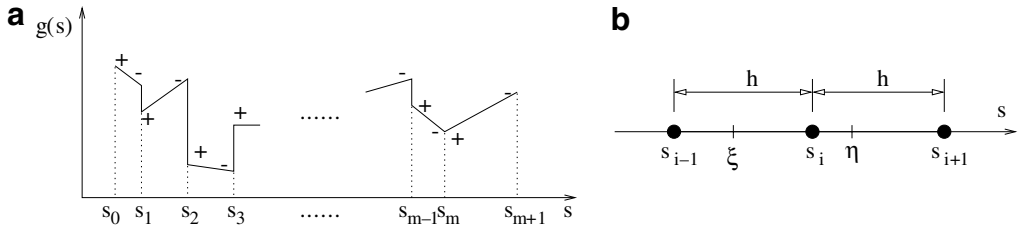


Fig. 2. Examples for generalized Taylor expansion and finite differences: (a) a non-smooth and discontinuous function, (b) a finite difference stencil with discontinuities.

Please cite this article in press as: S. Xu, Z. Jane Wang, A 3D immersed interface method for fluid–solid interaction, Comput. Methods Appl. Mech. Engrg. (2007), doi:10.1016/j.cma.2007.06.012

of a fluid with moving boundaries in 2D [33,1]. Simulation results indicate that the 2D immersed interface method (1) achieves near second-order accuracy in the infinity norm for both velocity and pressure, (2) introduces relatively insignificant cost with the addition of a solid in a simulation, and (3) conserves volumes enclosed by non-penetration boundaries.

In this paper, the derived jump conditions are used in an immersed interface method to simulate fluid–solid interaction in 3D. Compared with the existing 3D immersed boundary method, the current method has the improvements on spatial accuracy and resolution. It achieves near second-order accuracy in the infinity norm for the velocity, the accuracy for the pressure is between first and second order, and it does not smear sharp fluid–solid interfaces. Compared with body-fitted grid methods, the current method has the advantage in efficiency for moving boundary problems. Because of the use of a fixed Cartesian grid for fluids and Lagrangian markers for moving boundaries, the method does not need costly 3D grid regeneration, which is required in body-fitted grid methods. As shown in Section 6, the cost count of the current method in each time step is $\mathcal{O}(N\ln N)+\mathcal{O}(M\ln M)+\mathcal{O}(N)+\mathcal{O}(M)$, where N is the total number of Cartesian grid nodes and M is the total number of Lagrangian markers.

This paper is organized as follows. In Section 2, an overview of the method is given, which summarizes the major components needed by the method. Each major component is then presented in following sections. In Section 3, linear systems to determine jump conditions are listed. These jump conditions are the necessary ones to be incorporated into finite difference schemes. In Section 4, parametric triangulation of an interface is introduced. The parametric triangulation is used to identify finite difference stencils which pass across a fluid–solid interface. In Section 5, the interpolation of the velocity on staggered grid nodes and Lagrangian markers is presented. In Section 6, the major procedures of the current method are listed along with their cost counts. In Section 7, numerical examples are given to demonstrate the accuracy, conservation, and efficiency of the method. Last, Section 8 concludes the paper.

2. Overview of the method

Taking the divergence of the momentum equation, Eq. (1), the pressure Poisson equation is obtained, which reads

$$\Delta p = -\left(\frac{\partial D}{\partial t}+\nabla\cdot(2\mathbf{v}D)-\frac{1}{Re}\Delta D\right)+s_p + \nabla\cdot\left(\int_B \mathbf{f}(\alpha_1,\alpha_2,t)\delta(\mathbf{x}-\mathbf{X}(\alpha_1,\alpha_2,t))\,\mathrm{d}\alpha_1\,\mathrm{d}\alpha_2\right), \quad (7)$$

where $D=\nabla\cdot\mathbf{v}$ is the divergence of the velocity, and s_p is

$$s_p = 2\left(\frac{\partial u}{\partial x}\frac{\partial v}{\partial y}-\frac{\partial u}{\partial y}\frac{\partial v}{\partial x}+\frac{\partial u}{\partial x}\frac{\partial w}{\partial z}-\frac{\partial u}{\partial z}\frac{\partial w}{\partial x}+\frac{\partial v}{\partial y}\frac{\partial w}{\partial z}-\frac{\partial v}{\partial z}\frac{\partial w}{\partial y}\right). \quad (8)$$

Terms with the divergence D are kept in Eq. (7) to better enforce the divergence-free condition, and $\frac{\partial D}{\partial t}$ is discretized by assuming $D=0$ at the next time level.

The current method solves the momentum equation, Eq. (1), and the pressure Poisson equation, Eq. (7), using the MAC scheme, the fourth-order Runge–Kutta temporal integration, and an FFT-based Poisson solver. A MAC grid is a staggered Cartesian grid, on which the pressure p and the velocity components u, v, and w are arranged as in Fig. 3. Define the central finite difference operators δ_x, δ_y, δ_z, δ_{xx}, δ_{yy}, and δ_{zz} as

$$\delta_x(\cdot)_{i,j,k} = \frac{(\cdot)_{i+\frac{1}{2},j,k}-(\cdot)_{i-\frac{1}{2},j,k}}{\Delta x}+c_x(\cdot)_{i,j,k}, \quad (9)$$

$$\delta_y(\cdot)_{i,j,k} = \frac{(\cdot)_{i,j+\frac{1}{2},k}-(\cdot)_{i,j-\frac{1}{2},k}}{\Delta y}+c_y(\cdot)_{i,j,k}, \quad (10)$$

$$\delta_z(\cdot)_{i,j,k} = \frac{(\cdot)_{i,j,k+\frac{1}{2}}-(\cdot)_{i,j,k-\frac{1}{2}}}{\Delta z}+c_z(\cdot)_{i,j,k}, \quad (11)$$

$$\delta_{xx}(\cdot)_{i,j,k} = \frac{(\cdot)_{i+1,j,k}-2(\cdot)_{i,j,k}+(\cdot)_{i-1,j,k}}{\Delta x^2}+c_{xx}(\cdot)_{i,j,k}, \quad (12)$$

$$\delta_{yy}(\cdot)_{i,j,k} = \frac{(\cdot)_{i,j+1,k}-2(\cdot)_{i,j,k}+(\cdot)_{i,j-1,k}}{\Delta y^2}+c_{yy}(\cdot)_{i,j,k}, \quad (13)$$

$$\delta_{zz}(\cdot)_{i,j,k} = \frac{(\cdot)_{i,j,k+1}-2(\cdot)_{i,j,k}+(\cdot)_{i,j,k-1}}{\Delta z^2}+c_{zz}(\cdot)_{i,j,k}, \quad (14)$$

where Δx, Δy, Δz are the spatial steps as shown in Fig. 3, and c_x, c_y, c_z, c_{xx}, c_{yy}, and c_{zz} are jump contributions. If the stencils of the above finite difference operators do not cross any fluid–solid interface, the jump contributions are zero, and usual central finite difference schemes are recovered. If the stencils of the above finite difference operators cross a fluid–solid interface, the jump contributions are non-zero, and they can be calculated according to Eqs. (4) and (5).

With these central finite difference operators, the momentum equation, Eq. (1), and the pressure Poisson equation, Eq. (7), are spatially discretized as follows. The

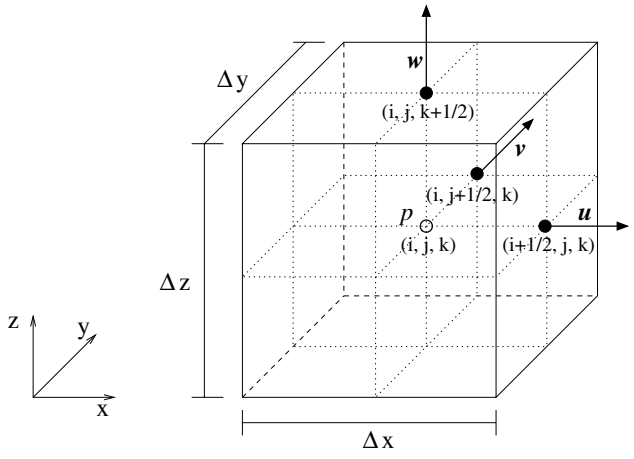


Fig. 3. Arrangements of the velocity components and the pressure on an MAC grid.

spatially discretized momentum equation for the velocity component u at $\left(i+\frac{1}{2},j,k\right)$ can be written as

$$\frac{\partial u}{\partial t} = -\delta_x(uu) - \delta_y(vu) - \delta_z(wu) - \delta_x p + \frac{1}{Re}(\delta_{xx}+\delta_{yy}+\delta_{zz})u, \tag{15}$$

where the subscript $\left(i+\frac{1}{2},j,k\right)$ is neglected in the operators. The similar equations for v at $\left(i,j+\frac{1}{2},k\right)$ and w at $\left(i,j,k+\frac{1}{2}\right)$ can be obtained. The spatially discretized pressure Poisson equation at (i,j,k) can be written as

$$(\delta_{xx}+\delta_{yy}+\delta_{zz})p = -\frac{\partial D}{\partial t} - 2(\delta_x(uD)+\delta_y(vD) + \delta_z(wD)) + \frac{1}{Re}(\delta_{xx}+\delta_{yy}+\delta_{zz})D + s_p^*, \tag{16}$$

where s_p^* is calculated at (i,j,k) as

$$s_p^* = 2(\delta_x u \delta_y v - \delta_y u \delta_x v + \delta_x u \delta_z w - \delta_z u \delta_x w + \delta_y v \delta_z w - \delta_z v \delta_y w). \tag{17}$$

Eqs. (15)–(17) and the discretized equations for v and w needs the values of the velocity components at the grid nodes with subscripts listed in Table 1. They can be interpolated from $u_{i+\frac{1}{2},j,k}$, $v_{i,j+\frac{1}{2},k}$, and $w_{i,j,k+\frac{1}{2}}$. The interpolation schemes are given in Section 5.

The RK4 temporal integration is used to march Eq. (15) and the spatially discretized equations for v and w in time. The reason to choose an explicit scheme is to be consistent with the explicit treatment of the motions of fluid–solid interfaces and the singular forces on the interfaces, which are functions of interface configurations. The reason to choose a high order scheme is to ensure numerical stability in the flow regime of moderate Reynolds numbers considered here. As pointed out by Johnston and Liu [9,10] and Weinan and Liu [7], high order explicit schemes are appropriate for flows of moderate to high Reynolds numbers, where viscous time step constraint is less restrictive than the convective one. The stability region of the RK4 scheme includes a portion of the imaginary axis, which ensures numerical stability of the background flow solver for the current flow regime.

Eq. (16) is a discretized Poisson equation as $\frac{\partial D}{\partial t}$ is approximated by assuming $D=0$ at the next time level, and it is solved in each substep of the RK4 temporal integration, as shown in [33]. Since the MAC grid is uniform in the current method, an FFT-based Poisson solver is adopted. The FFT-based Poisson solver can handle periodic boundary conditions and inhomogeneous Dirichlet, Neumann, and mixed boundary conditions by using FFT, sine, cosine, and quarter wave transformations, respectively [24].

Table 1
Subscripts of MAC grid nodes where velocity components need to be interpolated

u	i,j,k	$i+\frac{1}{2},j+\frac{1}{2},k$	$i+\frac{1}{2},j,k+\frac{1}{2}$	$i,j+\frac{1}{2},k$	$i,j,k+\frac{1}{2}$
v	i,j,k	$i+\frac{1}{2},j+\frac{1}{2},k$	$i,j+\frac{1}{2},k+\frac{1}{2}$	$i+\frac{1}{2},j,k$	$i,j,k+\frac{1}{2}$
w	i,j,k	$i+\frac{1}{2},j,k+\frac{1}{2}$	$i,j+\frac{1}{2},k+\frac{1}{2}$	$i+\frac{1}{2},j,k$	$i,j+\frac{1}{2},k$

A summary of the major components required by the current method can be given below.

- As indicated by Eqs. (4)–(6), necessary jump conditions are needed to obtain jump contributions in finite difference and interpolation schemes for Eqs. (15)–(17) and the spatially discretized equations for v and w.
- Jump contributions are non-zero only if the stencils of finite difference and interpolation schemes cross fluid–solid interfaces. In order to distinguish these stencils, the intersections between MAC grid lines and the interfaces need to be identified, including the coordinates of the intersections and the necessary jump conditions at the intersections.
- A fluid–solid interface follows the motion of the surrounding fluid. The fluid velocity is solved on MAC grid nodes, but the location of the interface is updated using the velocity of Lagrangian markers distributed on the interface. The velocity of Lagrangian markers needs to be interpolated from surrounding MAC grid nodes.

3. Computing the jump conditions

The derivation of jump conditions listed in this section can be found in [32]. The formulas for the jump conditions in this section are different from those in [32], but they are equivalent mathematically. The formulas given in this section are more amenable to numerical implementation.

The tangent vectors $\boldsymbol{\tau}$ and $\mathbf{b}$, and the normal vector $\mathbf{n}$ shown in Fig. 1 appear in the expressions for the jump conditions below. They are defined as follows:

$$\boldsymbol{\tau} = (\tau_1,\tau_2,\tau_3) = \frac{\partial \mathbf{X}}{\partial \alpha_1} = \left(\frac{\partial X}{\partial \alpha_1},\frac{\partial Y}{\partial \alpha_1},\frac{\partial Z}{\partial \alpha_1}\right), \tag{18}$$

$$\mathbf{b} = (b_1,b_2,b_3) = \frac{\partial \mathbf{X}}{\partial \alpha_2} = \left(\frac{\partial X}{\partial \alpha_2},\frac{\partial Y}{\partial \alpha_2},\frac{\partial Z}{\partial \alpha_2}\right), \tag{19}$$

$$\mathbf{n} = (n_1,n_2,n_3) = \boldsymbol{\tau} \times \mathbf{b}. \tag{20}$$

The parameters α_1 and α_2 are chosen such that the vector $\mathbf{n}$ points to outside a solid. In addition, the following definitions are used:

$$J = \|\mathbf{n}\|, \tag{21}$$

$$\mathbf{n}^* = \frac{\mathbf{n}}{J}, \tag{22}$$

$$\mathbf{F} = \frac{\mathbf{f}}{J}, \tag{23}$$

$$F_n = \mathbf{F}\cdot\mathbf{n}^*, \tag{24}$$

$$\mathbf{F}_\tau = \mathbf{F} - F_n\mathbf{n}^*, \tag{25}$$

where $\mathbf{n}^*$ is the unit normal vector, and $\mathbf{F}$ is the density of singular force in the Cartesian space. In the numerical examples presented in Section 7, the force density $\mathbf{F}$ is calculated based on force models that relate the force density $\mathbf{F}$ to the configuration of the Lagrangian markers. The

force models are given in each numerical example in Section 7.

The jump conditions for the velocity and the pressure are

$$[\mathbf{v}] = 0, \tag{26}$$

$$[p] = F_n. \tag{27}$$

The jump conditions for the first derivatives of the velocity satisfy

$$C_1 \begin{bmatrix} \frac{\partial \mathbf{v}}{\partial x} \\ \frac{\partial \mathbf{v}}{\partial y} \\ \frac{\partial \mathbf{v}}{\partial z} \end{bmatrix} = \begin{pmatrix} 0 \\ 0 \\ -ReJ\mathbf{F}_\tau \end{pmatrix}, \tag{28}$$

where $[\cdot]$ denotes jump conditions along the direction of $\mathbf{n}$, and the coefficient matrix C_1 is

$$C_1 = \begin{pmatrix} \tau_1 & \tau_2 & \tau_3 \\ b_1 & b_2 & b_3 \\ n_1 & n_2 & n_3 \end{pmatrix}. \tag{29}$$

The jump conditions for the first derivatives of the pressure satisfy

$$C_1 \begin{bmatrix} \frac{\partial p}{\partial x} \\ \frac{\partial p}{\partial y} \\ \frac{\partial p}{\partial z} \end{bmatrix} = \begin{pmatrix} \frac{\partial F_n}{\partial \alpha_1} \\ \frac{\partial F_n}{\partial \alpha_2} \\ \frac{\partial \tilde{f}_1}{\partial \alpha_1} + \frac{\partial \tilde{f}_2}{\partial \alpha_2} \end{pmatrix}, \tag{30}$$

where the contravariant components $\tilde{f}_1$ and $\tilde{f}_2$ in the parameter space are calculated by

$$\tilde{f}_1 = (\mathbf{b} \times \mathbf{n}^*) \cdot \mathbf{F}, \tag{31}$$

$$\tilde{f}_2 = (\mathbf{n}^* \times \boldsymbol{\tau}) \cdot \mathbf{F}. \tag{32}$$

The jump conditions for the second derivatives of the velocity satisfy

$$C_2 \begin{bmatrix} \frac{\partial^2 \mathbf{v}}{\partial x \partial x} \\ \frac{\partial^2 \mathbf{v}}{\partial x \partial y} \\ \frac{\partial^2 \mathbf{v}}{\partial x \partial z} \\ \frac{\partial^2 \mathbf{v}}{\partial y \partial y} \\ \frac{\partial^2 \mathbf{v}}{\partial y \partial z} \\ \frac{\partial^2 \mathbf{v}}{\partial z \partial z} \end{bmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -Re\frac{\partial J\mathbf{F}_\tau}{\partial \alpha_1} \\ -Re\frac{\partial J\mathbf{F}_\tau}{\partial \alpha_2} \\ Re[\nabla p] \end{pmatrix} - \begin{pmatrix} \frac{\partial^2 X}{\partial \alpha_1 \partial \alpha_1} & \frac{\partial^2 Y}{\partial \alpha_1 \partial \alpha_1} & \frac{\partial^2 Z}{\partial \alpha_1 \partial \alpha_1} \\ \frac{\partial^2 X}{\partial \alpha_2 \partial \alpha_2} & \frac{\partial^2 Y}{\partial \alpha_2 \partial \alpha_2} & \frac{\partial^2 Z}{\partial \alpha_2 \partial \alpha_2} \\ \frac{\partial^2 X}{\partial \alpha_1 \partial \alpha_2} & \frac{\partial^2 Y}{\partial \alpha_1 \partial \alpha_2} & \frac{\partial^2 Z}{\partial \alpha_1 \partial \alpha_2} \\ \frac{\partial n_1}{\partial \alpha_1} & \frac{\partial n_2}{\partial \alpha_1} & \frac{\partial n_3}{\partial \alpha_1} \\ \frac{\partial n_1}{\partial \alpha_2} & \frac{\partial n_2}{\partial \alpha_2} & \frac{\partial n_3}{\partial \alpha_2} \\ 0 & 0 & 0 \end{pmatrix} \begin{bmatrix} \frac{\partial \mathbf{v}}{\partial x} \\ \frac{\partial \mathbf{v}}{\partial y} \\ \frac{\partial \mathbf{v}}{\partial z} \end{bmatrix}, \tag{33}$$

where the coefficient matrix C_2 is

$$C_2 = \begin{pmatrix} \tau_1\tau_1 & \tau_1\tau_2+\tau_2\tau_1 & \tau_1\tau_3+\tau_3\tau_1 & \tau_2\tau_2 & \tau_2\tau_3+\tau_3\tau_2 & \tau_3\tau_3 \\ b_1b_1 & b_1b_2+b_2b_1 & b_1b_3+b_3b_1 & b_2b_2 & b_2b_3+b_3b_2 & b_3b_3 \\ \tau_1b_1 & \tau_1b_2+\tau_2b_1 & \tau_1b_3+\tau_3b_1 & \tau_2b_2 & \tau_2b_3+\tau_3b_2 & \tau_3b_3 \\ \tau_1n_1 & \tau_1n_2+\tau_2n_1 & \tau_1n_3+\tau_3n_1 & \tau_2n_2 & \tau_2n_3+\tau_3n_2 & \tau_3n_3 \\ b_1n_1 & b_1n_2+b_2n_1 & b_1n_3+b_3n_1 & b_2n_2 & b_2n_3+b_3n_2 & b_3n_3 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}, \tag{34}$$

and $\frac{\partial \mathbf{n}}{\partial \alpha_1} = \left(\frac{\partial n_1}{\partial \alpha_1}, \frac{\partial n_2}{\partial \alpha_1}, \frac{\partial n_3}{\partial \alpha_1}\right)$ and $\frac{\partial \mathbf{n}}{\partial \alpha_2} = \left(\frac{\partial n_1}{\partial \alpha_2}, \frac{\partial n_2}{\partial \alpha_2}, \frac{\partial n_3}{\partial \alpha_2}\right)$ are calculated numerically according to

$$\frac{\partial \mathbf{n}}{\partial \alpha_1} = \frac{\partial^2 \mathbf{X}}{\partial \alpha_1 \partial \alpha_1} \times \mathbf{b} + \boldsymbol{\tau} \times \frac{\partial^2 \mathbf{X}}{\partial \alpha_1 \partial \alpha_2}, \tag{35}$$

$$\frac{\partial \mathbf{n}}{\partial \alpha_2} = \frac{\partial^2 \mathbf{X}}{\partial \alpha_1 \partial \alpha_2} \times \mathbf{b} + \boldsymbol{\tau} \times \frac{\partial^2 \mathbf{X}}{\partial \alpha_2 \partial \alpha_2}. \tag{36}$$

The jump conditions for the second derivatives of the pressure satisfy

$$C_2 \begin{bmatrix} \frac{\partial^2 p}{\partial x \partial x} \\ \frac{\partial^2 p}{\partial x \partial y} \\ \frac{\partial^2 p}{\partial x \partial z} \\ \frac{\partial^2 p}{\partial y \partial y} \\ \frac{\partial^2 p}{\partial y \partial z} \\ \frac{\partial^2 p}{\partial z \partial z} \end{bmatrix} = \begin{pmatrix} \frac{\partial^2 F_n}{\partial \alpha_1 \partial \alpha_1} \\ \frac{\partial^2 F_n}{\partial \alpha_2 \partial \alpha_2} \\ \frac{\partial^2 F_n}{\partial \alpha_1 \partial \alpha_2} \\ \frac{\partial}{\partial \alpha_1}\left(\frac{\partial \tilde{f}_1}{\partial \alpha_1} + \frac{\partial \tilde{f}_2}{\partial \alpha_2}\right) \\ \frac{\partial}{\partial \alpha_2}\left(\frac{\partial \tilde{f}_1}{\partial \alpha_1} + \frac{\partial \tilde{f}_2}{\partial \alpha_2}\right) \\ [\nabla \cdot (\mathbf{v} \cdot \nabla \mathbf{v})] \end{pmatrix} - \begin{pmatrix} \frac{\partial^2 X}{\partial \alpha_1 \partial \alpha_1} & \frac{\partial^2 Y}{\partial \alpha_1 \partial \alpha_1} & \frac{\partial^2 Z}{\partial \alpha_1 \partial \alpha_1} \\ \frac{\partial^2 X}{\partial \alpha_2 \partial \alpha_2} & \frac{\partial^2 Y}{\partial \alpha_2 \partial \alpha_2} & \frac{\partial^2 Z}{\partial \alpha_2 \partial \alpha_2} \\ \frac{\partial^2 X}{\partial \alpha_1 \partial \alpha_2} & \frac{\partial^2 Y}{\partial \alpha_1 \partial \alpha_2} & \frac{\partial^2 Z}{\partial \alpha_1 \partial \alpha_2} \\ \frac{\partial n_1}{\partial \alpha_1} & \frac{\partial n_2}{\partial \alpha_1} & \frac{\partial n_3}{\partial \alpha_1} \\ \frac{\partial n_1}{\partial \alpha_2} & \frac{\partial n_2}{\partial \alpha_2} & \frac{\partial n_3}{\partial \alpha_2} \\ 0 & 0 & 0 \end{pmatrix} \begin{bmatrix} \frac{\partial p}{\partial x} \\ \frac{\partial p}{\partial y} \\ \frac{\partial p}{\partial z} \end{bmatrix}. \tag{37}$$

The calculation of surface derivatives with respect to α_1 and α_2 is presented in Section 4. The non-singularity (except at the two poles of a spherical interface) of the coefficient matrices C_1 and C_2 is proved in [32]. The small linear systems given by Eqs. (28), (30), (33) and (37) can be solved analytically, as shown in Appendix .

A flow quantity at a fixed point in space may have a jump in terms of time when a fluid–solid interface crosses the point, and a temporal jump condition can be related to a corresponding spatial jump condition [32,33]. In simulation of viscous flow, the incorporation of temporal jump conditions in temporal discretization has negligible effect on simulation results [33]. In the current method, temporal jump conditions are not included.

4. Parametric triangulation of an interface

The current method considers immersed solids whose surfaces are smooth, orientable, and topologically equivalent to a sphere or a torus.

4.1. Interface parametrization

The parametrization of an ellipsoidal shape is

Please cite this article in press as: S. Xu, Z. Jane Wang, A 3D immersed interface method for fluid–solid interaction, Comput. Methods Appl. Mech. Engrg. (2007), doi:10.1016/j.cma.2007.06.012

$$X_s = a\sin(\alpha_1)\cos(\alpha_2), \tag{38}$$
$$Y_s = b\sin(\alpha_1)\sin(\alpha_2), \tag{39}$$
$$Z_s = c\cos(\alpha_1), \tag{40}$$

where the coordinates (X_s, Y_s, Z_s) are used to express the shape, a and b are the equatorial radii along the x- and y-axes, c is the polar radius along the z-axis, and $\alpha_1 \in [0, \pi]$ and $\alpha_2 \in [0, 2\pi]$. The coordinates (X, Y, Z) of an ellipsoidal fluid–solid interface are related to the shape coordinates (X_s, Y_s, Z_s) through translation and rotation transformations. The interface is spherical if $a = b = c$. A Lagrangian interface mesh as illustrated in Fig. 4a is generated with the following parameter discretization:

$$\alpha_{1m_1} = \frac{\Delta\alpha_1}{2} + m_1\Delta\alpha_1, \quad m_1 = 0, 1, \ldots, M_1, \tag{41}$$
$$\alpha_{2m_2} = m_2\Delta\alpha_2, \quad m_2 = 0, 1, \ldots, M_2, \tag{42}$$

where $\Delta\alpha_1 = \frac{\pi}{M_1+1}$, $\Delta\alpha_2 = \frac{2\pi}{M_2}$, and M_2 is an even integer. The interface is tracked by Lagrangian markers located at the nodes of the interface mesh. The integers M_1 and M_2 are chosen such that the maximum distance between two neighboring Lagrangian markers is about the spatial step of the background MAC grid. The jump conditions in Section 3 are calculated at the Lagrangian markers. The total number of the Lagrangian markers on the interface is $M = M_1(M_2 - 1)$. No Lagrangian markers locate at the two poles corresponding to $\alpha_1 = 0$ and $\alpha_1 = \pi$. At the two poles, $J = 0$, and many equations in Section 3 have J in denominators of fractions. Realizing that the force density in the Cartesian space, $\mathbf{F} = \frac{\mathbf{f}}{J}$, is finite at the two poles, it can be shown that these fractions are also finite at the two poles. The current choice of the Lagrangian interface mesh excludes the two poles, so the values of these fractions at the two poles are not needed. Otherwise numerical extrapolation has to be applied.

The ellipsoidal interface is periodic in α_2. The first derivatives with respect to α_2 in Section 3 are calculated numerically using periodic cubic splines. The second derivatives with respect to α_2 are calculated from their corresponding first derivatives also using periodic cubic splines. In order to use periodic cubic splines to calculate surface derivatives with respect to α_1, a closed smooth curve on the interface for each $\alpha_{2m_2} \in [0, \pi]$ is composed by two branches corresponding to values of α_{2m_2} and $\pi + \alpha_{2m_2}$, as demonstrated in Fig. 5a. So M_2 has to be even. It can be shown that the functions $J\mathbf{F}_\tau$, $\tilde{f}_1$, and $\frac{\partial \tilde{f}_1}{\partial \alpha_1} + \frac{\partial \tilde{f}_2}{\partial \alpha_2}$ are continuous at the two poles and smooth away from the two poles on this closed curve. All other functions to be differentiated with respect to α_1 in Section 3 are smooth on the curve. On the branch corresponding to $\alpha_2 = \alpha_{2m_2}$,

$$\alpha_1^* = \alpha_1, \tag{43}$$
$$\frac{\partial^n}{\partial \alpha_1^n} = \frac{\partial^n}{\partial \alpha_1^{*n}}, \tag{44}$$

where α_1^* is defined in Fig. 5a, and $n = 0, 1, 2, \ldots$ On the branch corresponding to $\alpha_2 = \pi + \alpha_{2m_2}$,

$$\alpha_1^* = 2\pi - \alpha_1, \tag{45}$$
$$\frac{\partial^{2n}}{\partial \alpha_1^{2n}} = \frac{\partial^{2n}}{\partial \alpha_1^{*2n}}, \tag{46}$$
$$\frac{\partial^{2n+1}}{\partial \alpha_1^{2n+1}} = -\frac{\partial^{2n+1}}{\partial \alpha_1^{*2n+1}}. \tag{47}$$

Thus, periodic cubic splines with respect to α_1^* can be used to calculate $\frac{\partial^n}{\partial \alpha_1^n}$ with the above transformations. The cost count to compute all the surface derivatives in Section 3 is $\mathcal{O}(M)$.

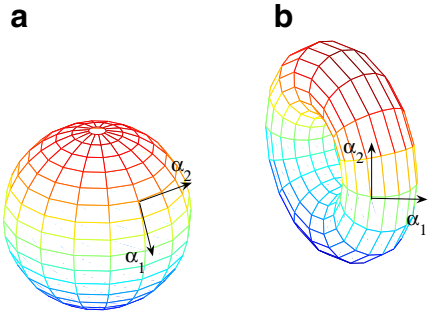


Fig. 4. Interface parametrization and interface meshes: (a) a sphere, (b) a torus.

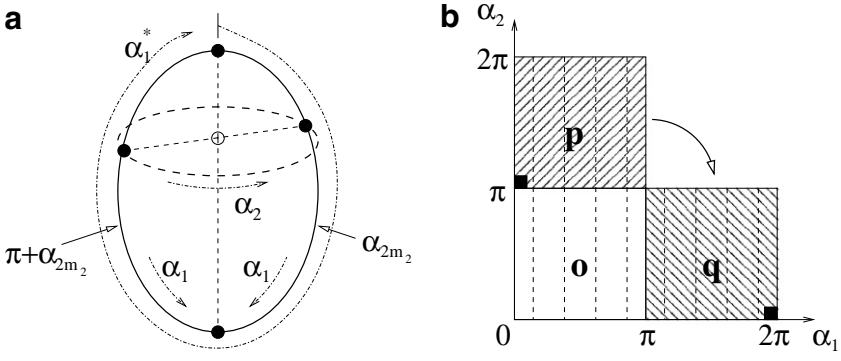


Fig. 5. Periodicity of an ellipsoidal interface: (a) formation of a closed curve past the two poles, (b) composition of periodic data.

Please cite this article in press as: S. Xu, Z. Jane Wang, A 3D immersed interface method for fluid–solid interaction, Comput. Methods Appl. Mech. Engrg. (2007), doi:10.1016/j.cma.2007.06.012

Because of the differentiation along the interface, it is important to maintain the smoothness of the interface for numerical stability. The current method employs Fourier filtering to smooth the interpolated velocity of Lagrangian markers on the interface. The interpolation approach is presented in Section 5. To filter in the direction of α_2, original data defined on the parameter regions "o" and "p" in Fig. 5b are directly used because of their periodicity in α_2. To filter in the direction of α_1, data defined on the parameter regions "o" and "q" in Fig. 5b are used, where data on the region "q" are obtained by re-organize the data on the region "p", as illustrated in Fig. 5b. It is required that α_1 start at $\frac{\Delta\alpha_1}{2}$ and end at $\pi - \frac{\Delta\alpha_1}{2}$. The cost count of Fourier filtering is $\mathcal{O}(M \ln M)$. The surface derivatives in Section 3 can also be computed using Fourier transformations with more cost than periodic cubic splines.

The parametrization of a torus is given by

$$X_s = r\cos(\alpha_1), \tag{48}$$

$$Y_s = (R + r\sin(\alpha_1))\cos(\alpha_2), \tag{49}$$

$$Z_s = (R + r\sin(\alpha_1))\sin(\alpha_2), \tag{50}$$

where R is the distance from the center of the torus tube to the center of the torus, r is the radius of the tube, and $\alpha_1 \in [0, 2\pi]$ and $\alpha_2 \in [0, 2\pi]$. A Lagrangian interface mesh as illustrated in Fig. 4b is generated with the following parameter discretization:

$$\alpha_{1m_1} = m_1\Delta\alpha_1, \quad m_1 = 0, 1, \ldots, M_1, \tag{51}$$

$$\alpha_{2m_2} = m_2\Delta\alpha_2, \quad m_2 = 0, 1, \ldots, M_2, \tag{52}$$

where $\Delta\alpha_1 = \frac{2\pi}{M_1}$ and $\Delta\alpha_2 = \frac{2\pi}{M_2}$. The total number of the Lagrangian markers on the torus is $M = (M_1 - 1)(M_2 - 1)$. The torus has periodicity in the both directions of α_1 and α_2. So implementation of periodic cubic splines to calculate surface derivatives and Fourier filtering to smooth the torus is straightforward.

4.2. Interface triangulation

As illustrated in Fig. 6, an interface are be approximated by small triangular patches formed from neighboring Lagrangian markers. In Fig. 6, two triangular patches, $\Delta P_1P_2P_4$ and $\Delta P_2P_3P_4$, are formed from four neighboring Lagrangian markers (nodes of the interface mesh): $P_1(\alpha_{1m_1}, \alpha_{2m_2})$, $P_2(\alpha_{1m_1} + \Delta\alpha_1, \alpha_{2m_2})$, $P_3(\alpha_{1m_1} + \Delta\alpha_1, \alpha_{2m_2} + \Delta\alpha_2)$, and $P_4(\alpha_{1m_1}, \alpha_{2m_2} + \Delta\alpha_2)$. The parametrization of an ellipsoidal interface leaves two holes at the two poles. The two holes are covered by triangular patches as illustrated for the hole at $\alpha_1 = 0$ in Fig. 7.

The intersections between an interface and MAC grid lines are found by projecting triangular patches along the x-, y-, and z-axes. Here the triangular patch $\Delta P_1P_2P_4$ in Fig. 8 is taken as an example. To find the intersections between this triangular patch and MAC grid lines parallel to the z-axis, $\Delta P_1P_2P_4$ is projected to the x–y plane along the z-axis to obtain the projection $\Delta Q_1Q_2Q_4$, which is contained inside a rectangle *I II III IV*. The rectangle is used to

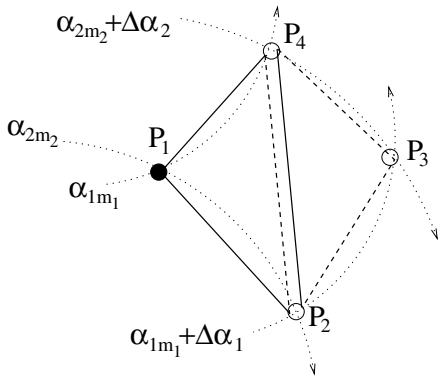


Fig. 6. Parametric triangulation of an interface.

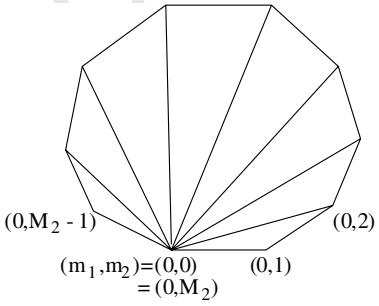


Fig. 7. Triangular patches covering the hole at a pole.

determine MAC grid lines which are parallel to the z-axis and may intersect $\Delta P_1P_2P_4$. If the projection $Q_X(x_I, y_J)$ of such a MAC grid line l, where $I = i$ or $i + \frac{1}{2}$ and $J = j$ or $j + \frac{1}{2}$, falls inside $\Delta Q_1Q_2Q_4$ (as the case in Fig. 8), on the edges of $\Delta Q_1Q_2Q_4$, or on the vertices of $\Delta Q_1Q_2Q_4$, the MAC grid line l intersects the triangular patch $\Delta P_1P_2P_4$ at the corresponding locations. The intersection is denoted as the point P_X in Fig. 8.

The x- and y-coordinates of the intersection P_X are (x_I, y_J). The z-coordinate and the necessary jump conditions at the intersection P_X are interpolated from the three vertices P_1, P_2, and P_4, where the values of the three Cartesian coordinates, the two Lagrangian parameters, and the necessary jump conditions are all known. If $\Delta P_1P_2P_4$ is not a triangular patch which covers the hole at a pole of an ellipsoidal interface, the following linear interpolation is used:

$$g(\alpha_1, \alpha_2) = c_{g0} + c_{g1}\alpha_1 + c_{g2}\alpha_2, \tag{53}$$

where g can be a Cartesian coordinate of an interface or a jump condition across an interface, and c_{g0}, c_{g1}, and c_{g2} are constants. The constants c_{g0}, c_{g1}, and c_{g2} are determined from the vertices in the following linear system:

$$g(\alpha_{1m_1}, \alpha_{2m_2}) = c_{g0} + c_{g1}\alpha_{1m_1} + c_{g2}\alpha_{2m_2}, \tag{54}$$

$$g(\alpha_{1m_1} + \Delta\alpha_1, \alpha_{2m_2}) = c_{g0} + c_{g1}(\alpha_{1m_1} + \Delta\alpha_1) + c_{g2}\alpha_{2m_2}, \tag{55}$$

$$g(\alpha_{1m_1}, \alpha_{2m_2} + \Delta\alpha_1) = c_{g0} + c_{g1}\alpha_{1m_1} + c_{g2}(\alpha_{2m_2} + \Delta\alpha_2). \tag{56}$$

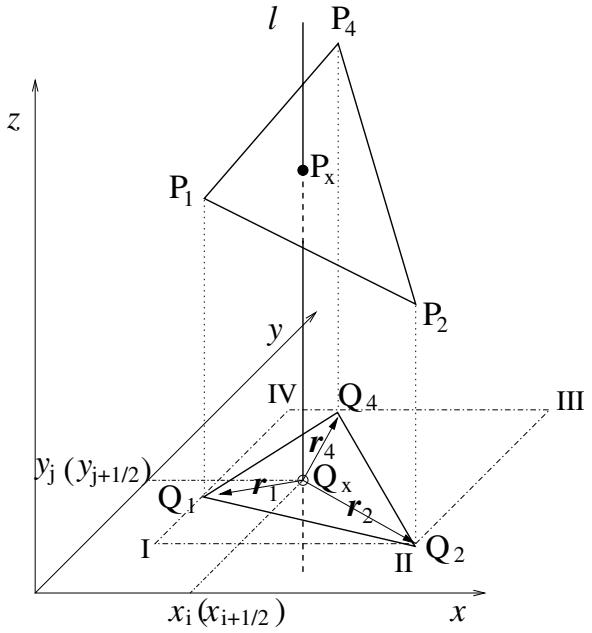


Fig. 8. Projection of a triangular patch for finding intersections between an interface and grid lines.

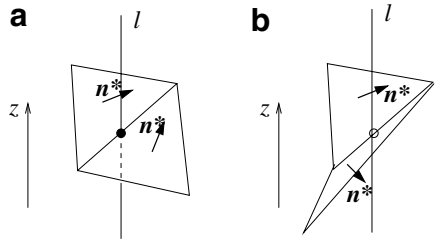


Fig. 9. Intersection on an edge: (a) recorded, (b) discarded.

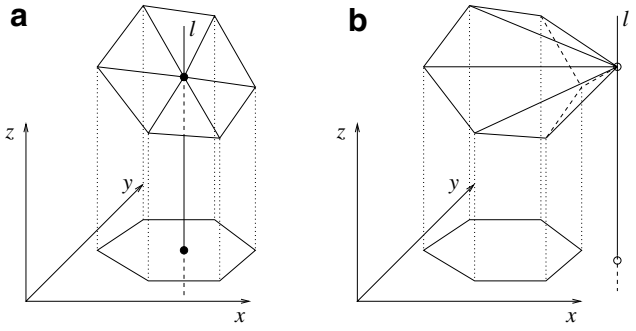


Fig. 10. Intersection on a vertex: (a) recorded, (b) discarded.

The values of the two parameters, α_1 and α_2, at the intersection P_X are determined from

$$x_I = c_{g0} + c_{g1}\alpha_1 + c_{g2}\alpha_2, \tag{57}$$

$$y_J = c_{g0} + c_{g1}\alpha_1 + c_{g2}\alpha_2. \tag{58}$$

This interpolation is of second-order accuracy in terms of $\Delta\alpha_1$ and $\Delta\alpha_2$. If $\Delta P_1P_2P_4$ is a triangular patch which covers the hole at a pole of an ellipsoidal interface, the following area-based interpolation formula is used instead:

$$g(P_X) = \frac{a_{g1}g(P_1) + a_{g2}g(P_2) + a_{g4}g(P_4)}{a_{g1} + a_{g2} + a_{g4}}, \tag{59}$$

where a_{g1}, a_{g2}, and a_{g4} are net areas calculated as follows:

$$a_{g1} = (\mathbf{r}_2 \times \mathbf{r}_4) \cdot \mathbf{e}_z, \tag{60}$$

$$a_{g2} = (\mathbf{r}_4 \times \mathbf{r}_1) \cdot \mathbf{e}_z, \tag{61}$$

$$a_{g4} = (\mathbf{r}_1 \times \mathbf{r}_2) \cdot \mathbf{e}_z, \tag{62}$$

where the vectors $\mathbf{r}_1$, $\mathbf{r}_2$, and $\mathbf{r}_4$ are defined in Fig. 8, and $\mathbf{e}_z$ is the unit basis vector of the z-axis.

After all the coordinates of the point P_X and all the necessary jump conditions at the point P_X are known, finite difference stencils which contain the point P_X are identified, and jump contributions to usual finite differences on these stencils are calculated. The above considers the situation in which MAC grid lines are parallel to the z-axis. A similar consideration applies to the other two directions. Before ending this section, some special situations which need special care are described below.

When an intersection falls on an edge (as in Fig. 9) or a vertex (as in Fig. 10) of a triangular patch, it is important to ensure that the intersection is not missed or repeated. First, the intersections falls on the edges represented by dashed lines or the vertices represented by open circles in Fig. 6 are not counted to avoid repetition. Second, if a MAC grid line is aligned with a triangular patch (the area of its projection is zero), the MAC grid line is regarded either parallel or tangential to the patch, and no intersection is recorded. Third, when an intersection falls on an edge but not a vertex of a triangular patch, it is counted only if the patch is oriented with respect to the corresponding MAC grid line in the same way as the adjacent patch, as illustrated in Fig. 9, where the orientation of the patches in Fig. 9 is defined as the sign of the z-component of the normal direction $\mathbf{n}^*$. Fourth, when the intersection is on a vertex, it is kept only if the projection of the vertex falls inside the polygon formed by the projection of all the triangular patches that share this common vertex, as illustrated in Fig. 10.

The equation of a projected edge can have different forms. For example the equation of the edge Q_1Q_2 in Fig. 8 can be written in the two forms as follows:

$$(y - Y_1)(X_2 - X_1) = (x - X_1)(Y_2 - Y_1), \tag{63}$$

$$(y - Y_2)(X_1 - X_2) = (x - X_2)(Y_1 - Y_2), \tag{64}$$

where (X_1, Y_1) and (X_2, Y_2) are the coordinates of the points Q_1 and Q_2, respectively. When determining whether the point Q_X falls inside $\Delta Q_1Q_2Q_4$, on the edge Q_1Q_2, or inside the neighboring triangle sharing the same edge Q_1Q_2, the same form of the equation of the edge Q_1Q_2 has to be used for the two adjacent triangles. Otherwise, the intersection P_X may be missed or repeated in counting due to rounding errors if the point Q_X falls on or is very close to the edge Q_1Q_2. This is a very trivial case, but it

Please cite this article in press as: S. Xu, Z. Jane Wang, A 3D immersed interface method for fluid–solid interaction, Comput. Methods Appl. Mech. Engrg. (2007), doi:10.1016/j.cma.2007.06.012

happens in practice. Finally, a crude check on intersection finding is the total number of intersections. It should be even.

An intersection on a triangular patch may coincide with a MAC grid node. If this occurs, the intersection is regarded at either one side or the other of the grid node along the x-, y-, and z-axes. When computing jump contributions to finite difference or interpolation schemes along each axis, a consistent choice can be made for this axis by infinitesimally detaching the triangular patch away from the grid node either toward the normal direction of the triangular patch or opposite to it.

5. Interpolation of the velocity

The staggered arrangement of the velocity components u, v and w and the pressure p, as illustrated in Fig. 3, necessitates the interpolation of the velocity components at the MAC grid nodes with subscripts listed in Table 1 from the defined velocity components $u_{i+\frac{1}{2},j,k}$, $v_{i,j+\frac{1}{2},k}$, and $w_{i,j,k+\frac{1}{2}}$. Define the interpolation operators ε_i, ε_j, ε_k as

$$\varepsilon_i(\cdot)_{i,j,k} = \frac{(\cdot)_{i+\frac{1}{2},j,k} + (\cdot)_{i-\frac{1}{2},j,k}}{2} + c_i(\cdot)_{i,j,k}, \tag{65}$$

$$\varepsilon_j(\cdot)_{i,j,k} = \frac{(\cdot)_{i,j+\frac{1}{2},k} + (\cdot)_{i,j-\frac{1}{2},k}}{2} + c_j(\cdot)_{i,j,k}, \tag{66}$$

$$\varepsilon_k(\cdot)_{i,j,k} = \frac{(\cdot)_{i,j,k+\frac{1}{2}} + (\cdot)_{i,j,k-\frac{1}{2}}}{2} + c_k(\cdot)_{i,j,k}, \tag{67}$$

where c_i, c_j, and c_k are jump contributions. If the stencils of the above interpolation operators do not cross any fluid–solid interface, the jump contributions are zero, and usual interpolation schemes are recovered. If the stencils of the above interpolation operators cross a fluid–solid interface, the jump contributions are non-zero, and they can be calculated according to Eq. (6). With these interpolation operators, the interpolation for the first row in Table 1 can be written as follows in the listing order:

$$u_{i,j,k} = \varepsilon_i u_{i,j,k}, \tag{68}$$

$$u_{i+\frac{1}{2},j+\frac{1}{2},k} = \varepsilon_j u_{i+\frac{1}{2},j+\frac{1}{2},k}, \tag{69}$$

$$u_{i+\frac{1}{2},j,k+\frac{1}{2}} = \varepsilon_k u_{i+\frac{1}{2},j,k+\frac{1}{2}}, \tag{70}$$

$$u_{i,j+\frac{1}{2},k} = \varepsilon_j u_{i,j+\frac{1}{2},k}, \tag{71}$$

$$u_{i,j,k+\frac{1}{2}} = \varepsilon_k u_{i,j,k+\frac{1}{2}}, \tag{72}$$

The interpolation for the second and the third rows in Table 1 can be written similarly.

A fluid–solid interface moves with the fluid. To update the interface location, the velocity of Lagrangian markers at the interface is interpolated from surrounding fluid velocity. The current method takes the interpolation strategy illustrated in Fig. 11. As shown in Fig. 11a, the velocity of the Lagrangian marker L on the interface is interpolated from two supplemental points N_+ and N_- along the normal direction $\mathbf{n}^*$ at the marker, and the two supplemental points are at the different sides of the interface with the equal distance Δn^* away from the interface. According to Eq. (6), the interpolation scheme at this step is

$$\mathbf{v}(L) = \frac{\mathbf{v}(N_+) + \mathbf{v}(N_-)}{2} - \frac{1}{2}\left[\frac{\partial \mathbf{v}(L)}{\partial n^*}\right]\Delta n^* + \mathcal{O}((\Delta n^*)^2), \tag{73}$$

where the normal derivative $\left[\frac{\partial \mathbf{v}}{\partial n^*}\right] = -Re\mathbf{F}_\tau$ according to Eq. (28). Trilinear interpolation is used to interpolate the velocity of each supplemental points from surrounding MAC grid points via transitional points in three separate steps, as shown in Fig. 11b, where the Cartesian grid points are the vertices of the cell, the transitional points lie on the edges and the faces of the cell, and a supplemental point locates inside the cell. The order of each interpolation step is marked in Fig. 11b. The distance Δn^* is chosen to be a little larger than $\sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2}$ such that all the interpolation cell in Fig. 11b is not cut by any interface. Thus a standard interpolation scheme can be applied in each interpolation step in Fig. 11b. Because of the staggered arrangement of the velocity components, the interpolation cell shown in Fig. 11b is different for the different velocity components. In Fig. 11b, if the cell is for the velocity component u, the transitional point II can be interpolated from the MAC grid points I and III as the following:

$$u(II) = \frac{z(III) - z(III)}{z(III) - z(I)}u(I) + \frac{z(II) - z(I)}{z(III) - z(I)}u(II) + \mathcal{O}((\Delta z)^2). \tag{74}$$

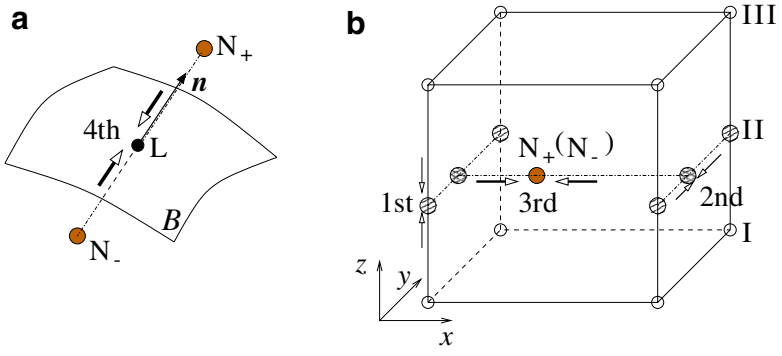


Fig. 11. Interpolation of the velocity of a Lagrangian marker.

More transit points and Cartesian grid nodes than in Fig. 11 can be used to achieve higher order interpolation as long as no interfaces cut interpolation cells.

There are other options available to interpolate the velocity of Lagrangian markers from surrounding fluid velocity. The following two options are avoided in the current practice. In the first one, the velocity of a Lagrangian marker is interpolated from the three closest intersections using a area-based formula similar to Eq. (59), and the velocity of the intersections is previously interpolated from MAC grid nodes both inside and outside of the interface. It turns out this option only gives first-order accuracy in the infinity norm for the velocity. In the second option, the velocity of a Lagrangian marker is extrapolated from MAC grid nodes either inside or outside of the interface. Accuracy of high order can be achieved, but the method suffers from a small numerical stability region with relatively low Reynolds numbers.

6. Summary of the method

The major procedures of the current method can now be summarized as follows. The computational cost associated with each step is also given, with N denoting the total number of MAC grid nodes for the pressure and M denoting the total number of Lagrangian markers. (Since the integers M_1 and M_2 are chosen such that the maximum distance between two neighboring Lagrangian markers is about the spatial step of the background MAC grid, the total number of intersections between MAC grid lines and fluid–solid interfaces is of order M too.)

1. Initializing the flow field and the interfaces $(\mathcal{O}(N) + \mathcal{O}(M))$;
2. calculating surface derivatives of geometric quantities $(\mathcal{O}(M))$;
3. modeling singular forces $(\mathcal{O}(M))$;
4. calculating surface derivatives of forcing quantities $(\mathcal{O}(M))$;
5. calculating jump conditions $(\mathcal{O}(M))$;
6. finding the intersections $(\mathcal{O}(M))$;
7. calculating jump contributions to finite difference and interpolation schemes $(\mathcal{O}(M))$;
8. interpolating and smoothing the velocity of the Lagrangian markers $(\mathcal{O}(M) + \mathcal{O}(M \ln M))$
9. updating the interface configurations $(\mathcal{O}(M))$;
10. solving the pressure field $(\mathcal{O}(N \ln N))$;
11. updating the velocity field $(\mathcal{O}(N))$.

7. Numerical examples

In this section, numerical examples simulated by the current method are given to test its accuracy, conservation, and efficiency.

7.1. Flow inside a rotating object

This first example considers the steady flow inside an object which rotates with a constant angular velocity Ω around a unit vector $\mathbf{R}$. The object is in the middle of a $[-1,1] \times [-1,1] \times [-1,1]$ cuboid. Rigid-wall boundary conditions are applied at the six sides of the cuboid. The analytical solution of the flow inside the rotating object is

$$\mathbf{v}(\mathbf{x}) = \Omega \mathbf{R} \times \mathbf{x}, \tag{75}$$

$$p(\mathbf{x}) = \frac{1}{2}\Omega^2 \|\mathbf{R} \times \mathbf{x}\|_2^2 + p_0, \tag{76}$$

where p_0 is an arbitrary constant. To enforce the prescribed rotation, each Lagrangian marker on the object is connected to its prescribed position by a linear spring, and the density of the singular force from the spring model is given by the following equation:

$$\mathbf{F} = K_s(\mathbf{X}_e - \mathbf{X}), \tag{77}$$

where K_s is the spring stiffness, and $\mathbf{X}_e$ is the prescribed position of a Lagrangian marker.

7.1.1. Spatial convergence

Spatial convergence of the method is indicated by the change of simulation errors with grid refinement. The simulation errors are based on the analytical solution, Eqs. (75) and (76). Table 2 presents the results of spatial convergence analysis for the steady flow inside a rotating sphere with the diameter equal to 1 at $Re = 10$. The angular velocity of the rotation is $\Omega = 1$, and the rotation direction is given by $\mathbf{R} = \left(\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}\right)$. The value of the spring stiffness in the spring model is $K_s = 1000$. In Table 2, N_x, N_y, and N_z are the numbers of MAC cells for the pressure p along the x-, y-, and z-axes, respectively. The order of accuracy is calculated by the following formula:

$$\text{order} = \frac{\ln(\|e_{\text{current}}\|_\infty / \|e_{\text{previous}}\|_\infty)}{\ln(\Delta_{\text{current}} / \Delta_{\text{previous}})}, \tag{78}$$

where e_{current} and e_{previous} denote the errors at the current and the previous rows in Table 2, respectively, and Δ_{current} and Δ_{previous} denote the corresponding spatial resolutions. The results in Table 2 indicate that the accuracy for the three velocity components u, v, and w is near second-order, and the accuracy for the pressure p is between first and second order.

Table 3 are the results for the steady flow inside a rotating torus with $R = 0.5$ and $r = 0.25$ at $Re = 10$. The angular velocity of the rotation is $\Omega = 1$, and the rotation direction is $\mathbf{R} = (1, 0, 0)$. The value of the spring stiffness is $K_s = 1000$. The results in Table 3 also indicate that the accuracy for the velocity components u, v and w is near second-order, and the accuracy for the pressure p is between first and second order.

7.1.2. Effect of spring stiffness

The effect of spring stiffness is investigated by simulating the steady flow inside a rotating sphere with different values

Table 2
Spatial convergence analysis for the flow inside a rotating sphere

$N_x \times N_y \times N_z$, $M_1 \times M_2$	$\|e_u\|_\infty$	Order	$\|e_v\|_\infty$	Order	$\|e_w\|_\infty$	Order	$\|e_p\|_\infty$	Order
$25\times25\times25$, 24×48	3.51×10^{-2}		3.45×10^{-2}		3.41×10^{-2}		2.33×10^{-2}	
$33\times33\times33$, 32×64	2.23×10^{-2}	1.63	2.21×10^{-2}	1.60	2.19×10^{-2}	1.60	1.15×10^{-2}	2.45
$49\times49\times49$, 48×96	1.12×10^{-2}	1.74	1.14×10^{-2}	1.67	1.12×10^{-2}	1.70	6.83×10^{-3}	1.28
$65\times65\times65$, 64×128	6.84×10^{-3}	1.75	6.88×10^{-3}	1.79	7.51×10^{-3}	1.41	5.40×10^{-3}	0.82
$97\times97\times97$, 96×192	3.29×10^{-3}	1.83	3.32×10^{-3}	1.82	3.29×10^{-3}	2.06	2.94×10^{-3}	1.52

Table 3
Spatial convergence analysis for the flow inside a rotating torus

$N_x \times N_y \times N_z$, $M_1 \times M_2$	$\|e_u\|_\infty$	Order	$\|e_v\|_\infty$	Order	$\|e_w\|_\infty$	Order	$\|e_p\|_\infty$	Order
$25\times25\times25$, 24×48	1.58×10^{-2}		1.12×10^{-1}		1.15×10^{-1}		1.42×10^{-1}	
$33\times33\times33$, 32×64	1.51×10^{-2}	0.16	8.17×10^{-2}	1.14	8.18×10^{-2}	1.23	9.23×10^{-2}	1.50
$49\times49\times49$, 48×96	8.16×10^{-3}	1.56	4.92×10^{-2}	1.28	4.81×10^{-2}	1.34	5.98×10^{-2}	1.07
$65\times65\times65$, 64×128	5.30×10^{-3}	1.53	2.10×10^{-3}	3.01	2.08×10^{-2}	2.97	3.56×10^{-2}	1.80
$97\times97\times97$, 96×192	2.97×10^{-3}	1.45	9.85×10^{-3}	1.89	9.68×10^{-3}	1.91	2.27×10^{-2}	1.11

of spring stiffness. The spatial resolution of the simulation corresponds to $N_x \times N_y \times N_z = 33\times33\times33$ and $M_1 \times M_2 = 32\times64$.

Table 4 lists the change of simulation errors against the spring stiffness. For this particular flow, the spring stiffness in the considered range has very small effect on the infinity norm of the simulation errors for both the velocity and the pressure. The errors for the positions of Lagrangian markers are plotted for $K_s = 10$ and $K_s = 5000$ in Fig. 12. The amplitudes of the position errors are much larger for much smaller spring stiffness, as expected.

The spring model introduces a vibration time scale into the flow. The effect of this time scale on a time-dependent flow has been investigated in [33]. How to choose the values of the spring stiffness has also been discussed in [33].

7.2. Flow induced by a relaxing balloon

In the second example, a 3D pressurized balloon immersed in an incompressible fluid relaxes to its spherical equilibrium shape from the initial distortion. The initial velocity and the pressure are set zero, and the coupled motion of the balloon and the fluid is driven only by balloon tension. The simulation domain is a square box with dimensions $[-0.8, 0.8]\times[-0.8, 0.8]\times[-0.8, 0.8]$. The initial shape of the balloon is an ellipsoid with $a = 0.64$, $b = 0.4$, and $c = 0.25$. The center coordinates of the ellipsoid are $(0, 0, 0)$. The density of singular force is modeled by

$$\mathbf{F} = EH\mathbf{n}^*, \tag{79}$$

where H is the mean curvature of the balloon surface, and $E = 0.2$ is a constant. At equilibrium, the balloon should be a sphere with radius $r_o = \sqrt[3]{abc} = 0.4$ and center coordinates $(0, 0, 0)$, the velocity should be zero everywhere, and the pressure should be piecewise constants with a jump, $[p] = -\frac{E}{r_o} = -0.5$, across the balloon surface.

Shown in Fig. 13 is the simulated evolution of balloon shape at $Re = 100$. The spatial resolution of this simulation is $N_x \times N_y \times N_z = 33\times33\times33$ and $M_1 \times M_2 = 32\times64$, and the time step of this simulation is fixed with $\Delta t = 0.01$. At this Reynolds number, the balloon undergoes a couple of oscillations before it settles down to equilibrium, as indicated in Fig. 14a. In Fig. 14a, the distances r_a, r_b, and r_c to the domain center $(0, 0, 0)$ from three Lagrangian points $(m_1, m_2) = \left(\frac{M_1}{2}, 0\right)$, $\left(\frac{M_1}{2}, \frac{M_2}{4}\right)$, and $(0, 0)$, are plotted against time at four different spatial simulation resolutions. The volume conservation of the balloon in the relaxation process is checked in Fig. 14b. The volume is well conserved during the process with its calculated values very close to the analytical one, $\frac{4\pi abc}{3} \approx 0.2681$. The volume errors during the relaxation process are of the same amplitudes as their initial values (at the time $t = 0$) which are due to the approximation of the volume by a Riemann sum. The relative volume errors do not exceed 0.37% at the four resolutions. The equilibrium pressure at two slices shown in Fig. 15 indicates that it is piecewise constants with a jump across the balloon surface.

The volume conservation is checked in Fig. 16 for $Re = 1$. The spatial resolution of this simulation is $N_x \times N_y \times N_z = 33\times33\times33$ and $M_1 \times M_2 = 32\times64$, and

Table 4
Effect of spring stiffness on the simulation accuracy for the flow inside a rotating sphere

K_s	10	100	500	1000	2000	5000
$\|e_u\|_\infty$	2.35×10^{-2}	2.25×10^{-2}	2.23×10^{-2}	2.23×10^{-2}	2.23×10^{-2}	2.25×10^{-2}
$\|e_v\|_\infty$	2.35×10^{-2}	2.26×10^{-2}	2.25×10^{-2}	2.21×10^{-2}	2.60×10^{-2}	2.25×10^{-2}
$\|e_w\|_\infty$	2.39×10^{-2}	2.22×10^{-2}	2.20×10^{-2}	2.19×10^{-2}	2.19×10^{-2}	2.19×10^{-2}
$\|e_p\|_\infty$	1.26×10^{-2}	1.61×10^{-2}	1.28×10^{-2}	1.15×10^{-2}	1.98×10^{-2}	1.44×10^{-2}

Please cite this article in press as: S. Xu, Z. Jane Wang, A 3D immersed interface method for fluid–solid interaction, Comput. Methods Appl. Mech. Engrg. (2007), doi:10.1016/j.cma.2007.06.012

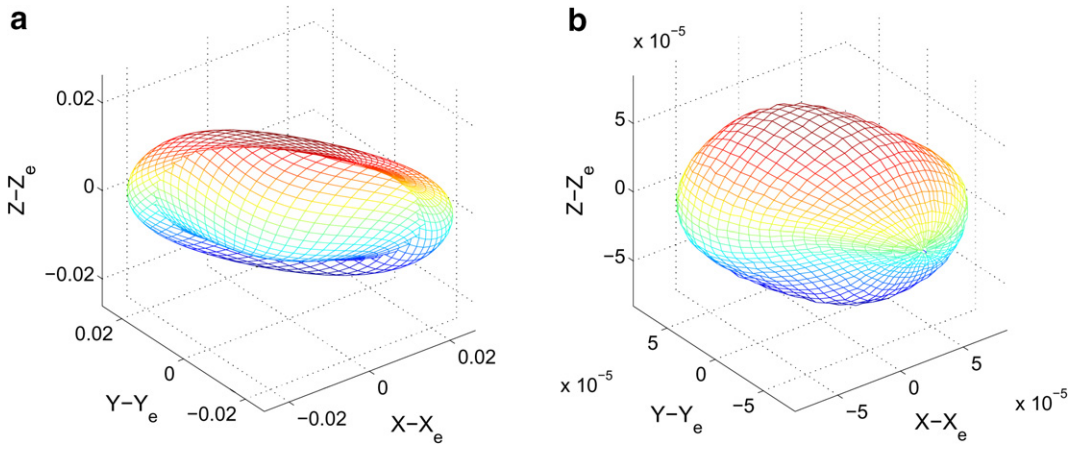


Fig. 12. Errors for the positions of Lagrangian markers: (a) $K_s = 10$, (b) $K_s = 5000$.

the time step $\Delta t = 0.005$ is also fixed. No oscillations are observed at this low Reynolds number, and it takes a very long time for the balloon to reach equilibrium.

Fig. 17 plots the distances r_a, r_b, and r_c, and the volume against time for $Re = 10$ at different CFL numbers. The convective and viscous CFL numbers CFL_c and CFL_μ are defined as

$$\mathrm{CFL}_c = \Delta t\left(\frac{u_{\max}}{\Delta x} + \frac{v_{\max}}{\Delta y} + \frac{w_{\max}}{\Delta z}\right), \tag{80}$$

$$\mathrm{CFL}_\mu = \frac{\Delta t}{Re}\left(\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} + \frac{1}{(\Delta z)^2}\right), \tag{81}$$

where $u_{\max}$, $v_{\max}$, and $w_{\max}$ are the maximum velocity components in the flow field. Again, the volume of the balloon is well preserved in the relaxation process, with the relative errors less than 0.52% for the four values of the CFL numbers.

7.3. Flow past a stationary sphere

Flow past a stationary sphere at varying Reynolds numbers, $Re = 10$, 20, 100, and 200, is simulated in this example. The spring model given by Eq. (77) is used to calculate the density of singular force with $K_s = 1000$, 500, 200, and 100 for $Re = 10$, 20, 100, and 200, respectively. The sphere locates in a box domain. Its diameter equals to 1, and its center coordinates are $(0,0,0)$. The spatial resolution of the simulation is given by $N_x \times N_y \times N_z = 256 \times 129 \times 129$ and $M_1 \times M_2 = 32 \times 64$. For $Re = 10$, 20, and 100, the domain sizes are $[-4,16] \times [-4,4] \times [-4,4]$ along the x-, y-, and z-axes, and the fixed time step $\Delta t = 0.005$ is used. For $Re = 200$, the domain sizes are $[-2,8] \times [-2,2] \times [-2,2]$, and the time step is controlled by $\mathrm{CFL}_c = \mathrm{CFL}_\mu = 0.2$. A free stream enters the domain in the direction of the x-axis. At the four sides of the domain, symmetric boundary conditions are used. At the domain outlet, the following outflow boundary conditions are used:

$$\frac{\partial \mathbf{v}}{\partial x} = 0, \tag{82}$$

$$\frac{\partial p}{\partial x} = \frac{1}{Re}\frac{\partial^2 u}{\partial x^2}. \tag{83}$$

At $Re = 10$, 20, and 100, the initial flow field is set uniform with $u = 1$ as the far field, and the sphere impulsively stops

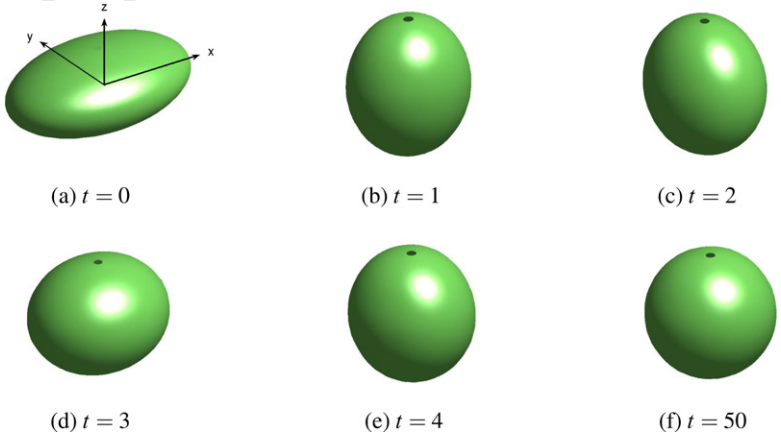


Fig. 13. Evolution of the balloon shape at $Re = 100$.

Please cite this article in press as: S. Xu, Z. Jane Wang, A 3D immersed interface method for fluid–solid interaction, Comput. Methods Appl. Mech. Engrg. (2007), doi:10.1016/j.cma.2007.06.012

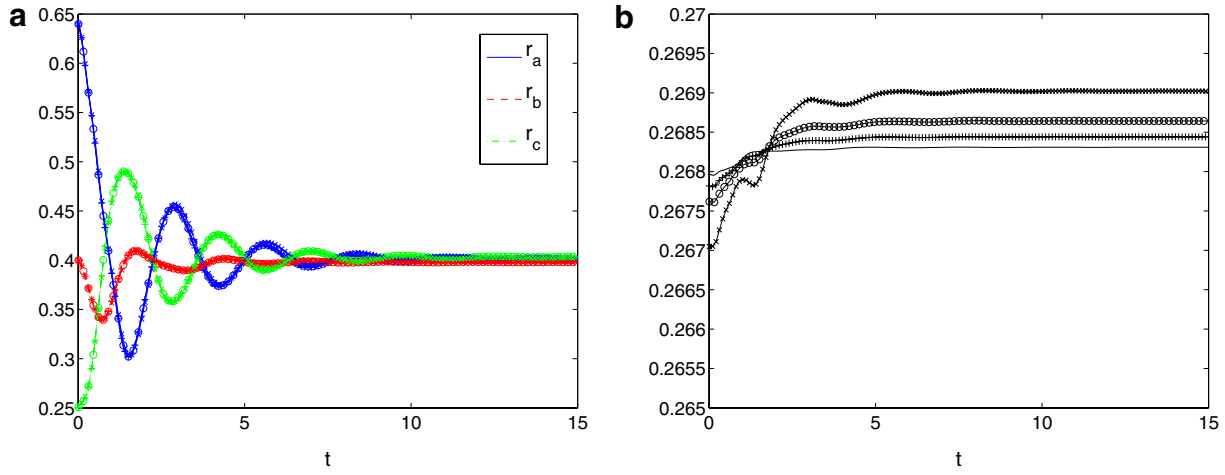


Fig. 14. Volume conservation at $Re = 100$ with different spatial resolutions: (a) temporal change of distances from three Lagrangian markers to the domain center, (b) temporal change of volume. $N_x \times N_y \times N_z$ and $M_1 \times M_2$: $97 \times 97 \times 97$ and 96×192 for lines, $65 \times 65 \times 65$ and 64×128 for lines with "+" marks, $49 \times 49 \times 49$ and 48×96 for lines with "o" marks, and $33 \times 33 \times 33$ and 32×64 for lines with "×" marks.

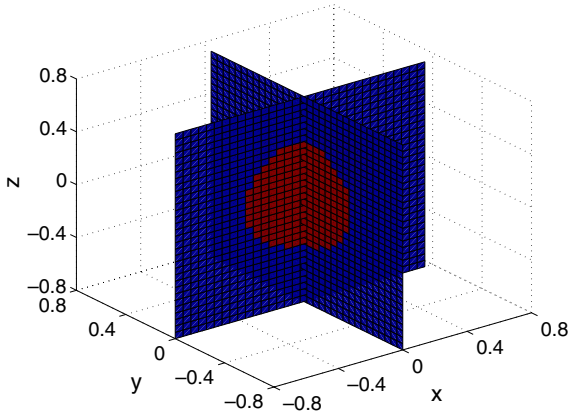


Fig. 15. Equilibrium pressure p at the two slices $x = 0$ and $y = 0$.

from the uniform velocity. At $Re = 200$, the initial flow field is set quiescent with $u = 0$, and the following formula is used to ramp the far-field uniform flow from $u = 0$ to $u = 1$ to avoid impulsive stop of the sphere

$$u_n = u_n + \exp\left(-\frac{t_{n-1}}{t_c}\right) - \exp\left(-\frac{t_n}{t_c}\right), \quad n = 1, 2, \ldots, \tag{84}$$

where the subscript n denotes a discrete time level, $t_0 = 0$ corresponds to the initial time, and $t_c = 1$ is a characteristic time of the ramping process. Correspondingly, the boundary condition for u at the domain inlet is $u = 1 - \exp\left(-\frac{t_n}{t_c}\right)$ instead of $u = 1$ at the other Reynolds numbers.

In general, the fluid force $\mathbf{G}$ applied by a fluid to an object can be calculated by

$$\mathbf{G} = \int_S \left(-p^+ \mathbf{n}^* + \frac{1}{Re}\left(\frac{\partial \mathbf{v}}{\partial n^*}\right)^+\right) \mathrm{d}S = -\int_S \mathbf{f}\, \mathrm{d}\alpha_1\, \mathrm{d}\alpha_2 + \int_S \left(-p^- \mathbf{n}^* + \frac{1}{Re}\left(\frac{\partial \mathbf{v}}{\partial n^*}\right)^-\right) \mathrm{d}S, \tag{85}$$

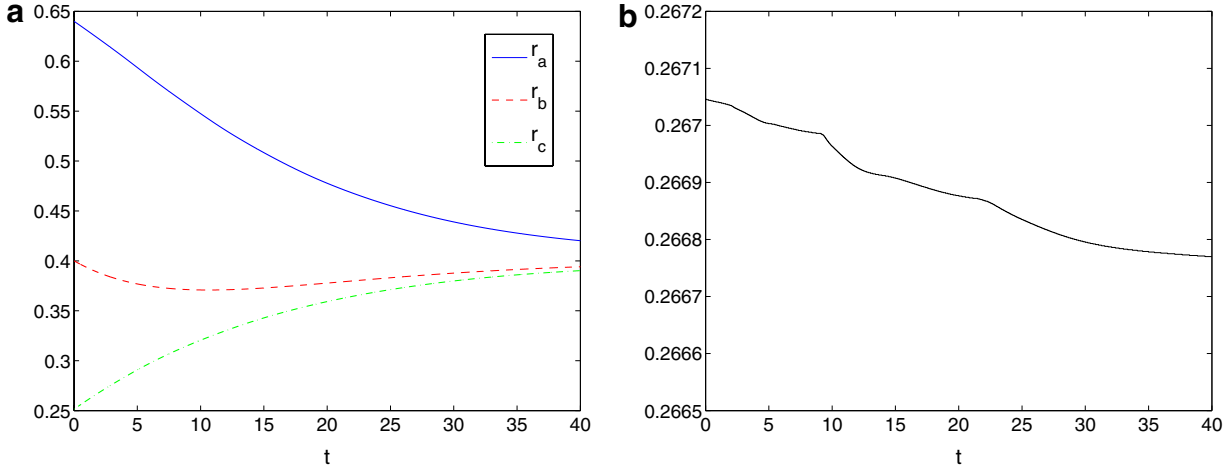


Fig. 16. Volume conservation at $Re = 1$: (a) temporal change of distances from three Lagrangian markers to the domain center, (b) temporal change of volume.

Please cite this article in press as: S. Xu, Z. Jane Wang, A 3D immersed interface method for fluid–solid interaction, Comput. Methods Appl. Mech. Engrg. (2007), doi:10.1016/j.cma.2007.06.012

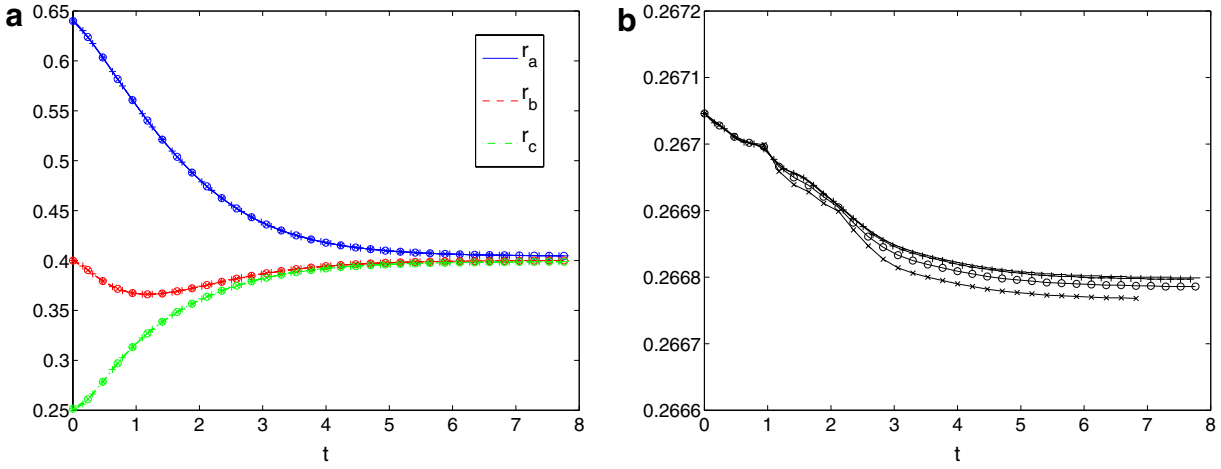


Fig. 17. Volume conservation at $Re = 10$ at different CFL numbers: (a) temporal change of distances from three Lagrangian markers to the domain center, (b) temporal change of volume. lines: $CFL_c = CFL_\mu = 0.05$, lines with "+" marks: $CFL_c = CFL_\mu = 0.1$, lines with "o" marks: $CFL_c = CFL_\mu = 0.3$, and lines with "×" marks: $CFL_c = CFL_\mu = 0.6$.

where the subscript "+" denotes the outer side of the object surface (fluid–solid interface) and the subscript "−" denotes the inner side. The term $\int_S \mathbf{f}\,d\alpha_1\,d\alpha_2$ is just the resultant external force on the object generated by a force model (a collection of springs in this example). For a centrally symmetric object, its motion can be regarded as the superposition of translation and rotation at its geometric center, and the following relations apply:

$$\int_S (-p^- \mathbf{n}^*)\mathrm{d}S = V\frac{\mathrm{d}\mathbf{U}_t}{\mathrm{d}t}, \tag{86}$$

$$\int_S \left(\frac{\partial \mathbf{v}}{\partial n^*}\right)^- \mathrm{d}S = 0, \tag{87}$$

where $\mathbf{U}_t$ is the translational velocity of the object, and V is the object volume. Thus the calculation of the fluid force can be simplified to

$$\mathbf{G} = -\int_S \mathbf{f}\,d\alpha_1\,d\alpha_2 + V\frac{\mathrm{d}\mathbf{U}_t}{\mathrm{d}t}, \tag{88}$$

which is simply the application of Newton's second law to the translational motion of the fluid contained inside the object. The sphere is stationary in the current example, so the calculation of the fluid force is given by

$$\mathbf{G} = -\int_S \mathbf{f}\,d\alpha_1\,d\alpha_2. \tag{89}$$

Fig. 18 shows the time history of the drag coefficient C_1 for flow past the sphere. The force coefficients C_1, C_2, and C_3 are defined as $(C_1, C_2, C_3) = \mathbf{C} = 2\mathbf{G}/A_f$, where $A_f = \frac{\pi}{4}$ is the projected frontal area for the sphere. A constant value of the drag coefficient is achieved for each of the Reynolds numbers as the flow becomes steady after the transit. Its value is compared with previous computational results [26,11] in Table 5, and the agreement is very good. Due to the spring model, oscillations in the drag coefficient are observed in the transit. With the ramping process described by Eq. (84), the amplitudes of the oscillations are significantly reduced, as indicated in Fig. 19.

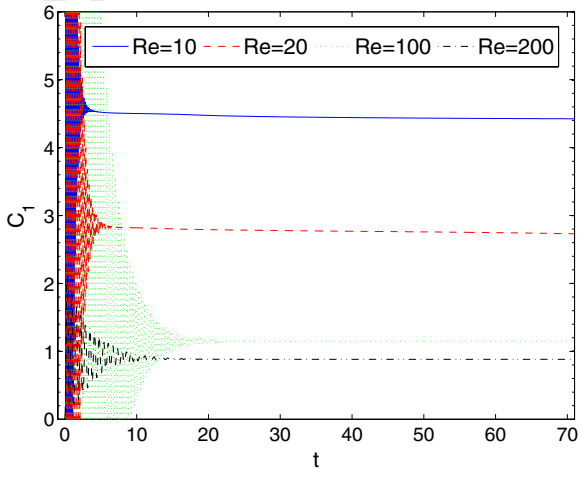


Fig. 18. Time history of the drag coefficient for flow past a stationary sphere at different Reynolds numbers.

Table 5
Drag coefficient and wake structure of flow past a sphere

	Re	10	20	100	200
C_1	Current	4.42	3.73	1.15	0.88
	Previous	4.4	3.8	1.2	0.8
Θ_s, L_{TB}	Current	–	–	130, 0.89	119, 1.44
	Previous	–	–	128, 0.90	116, 1.45
(x_{TB}, z_{TB})	Current	–	–	(0.76, 0.29)	(0.90, 0.36)
	previous	–	–	(0.76, 0.30)	(0.90, 0.36)

The distributions of the streamwise velocity u along the x-axis are plotted in Fig. 20. Negative values of u at $Re = 100$ and 200 indicate recirculating flow behind the sphere. No recirculating flow exists at $Re = 10$ and 20. Streamline plots in Fig. 21 confirm that recirculating flow exists at $Re = 100$ but not at $Re = 10$.

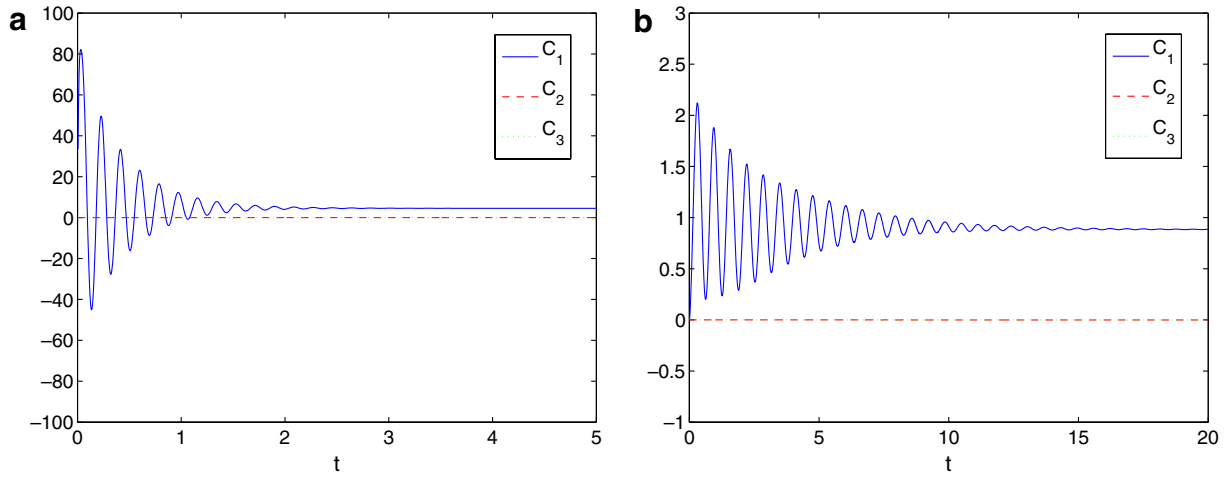


Fig. 19. Transient behavior of force coefficients for flow past a stationary sphere: (a) $Re = 10$ with no ramping process, (b) $Re = 200$ with a ramping process.

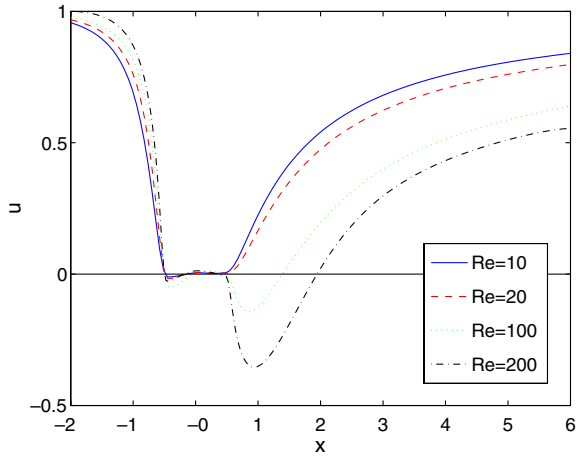


Fig. 20. The distribution of the streamwise velocity u along the x-axis.

The separation angle Θ_s, the length of a separation bubble L_{TB}, and the coordinates of a separation bubble center in the x–z plane (x_{TB}, z_{TB}) are illustrated in Fig. 21b. Their values from the current simulation are compared with previous computational results [26,11] in Table 5, and agree with the previous results very well.

In Fig. 22, the surface pressure p_s on the sphere is shown for $Re = 200$. The surface pressure is calculated by $p_s = \mathbf{F}_n$ subject to a constant. It is axisymmetric around the x-axis with the highest value at the stagnation point at this Reynolds number.

7.4. Flow around a flapper

In this example, flow around a hovering flapper is simulated. The spring model given by Eq. (77) with $K_s = 100$ is used to calculate the density of singular force. The flapper is an ellipsoid with $a = 0.4$, $b = 0.5$, and $c = 0.2$, and its two poles are located in the middle of flat surfaces. The flapper is contained in a rigid box of dimensions $[-2, 4] \times [-1, 1] \times [-2, 4]$. The simulation resolution is $N_x \times N_y \times N_z = 129 \times 65 \times 129$ and $M_1 \times M_2 = 32 \times 64$. The time step is fixed in this simulation, and it is $\Delta t = 1.96 \times 10^{-3} \approx \frac{T_f}{4000}$, where T_f is the flapping period. The motion of the flapper is formulated as

$$x_c = 1.25(\cos(0.8t) + 1)\cos\left(\frac{\pi}{3}\right), \tag{90}$$

$$y_c = 0, \tag{91}$$

$$z_c = 1.25(\cos(0.8t) + 1)\sin\left(\frac{\pi}{3}\right), \tag{92}$$

$$\theta = \frac{3\pi}{4} + \frac{\pi}{4}\sin(0.8t)\left(1 - \exp\left(-\frac{t}{t_c}\right)\right), \tag{93}$$

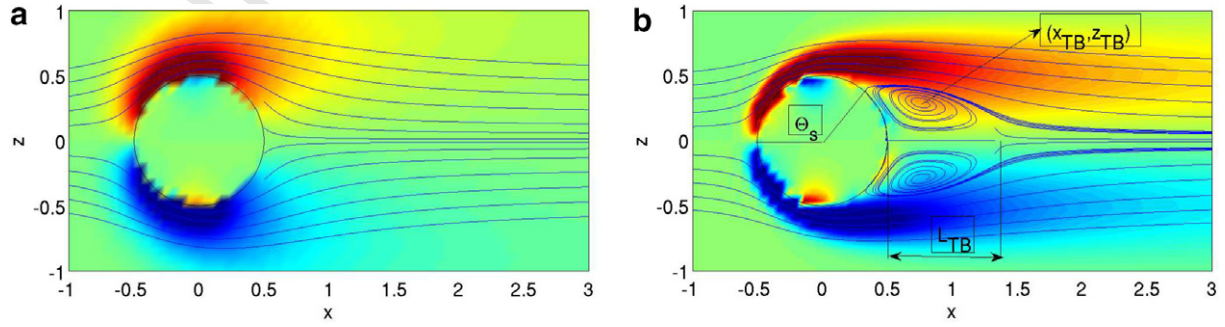


Fig. 21. Streamlines on the top of contours of the y-component of vorticity at the x–z plane: (a) $Re = 10$, (b) $Re = 100$.

Please cite this article in press as: S. Xu, Z. Jane Wang, A 3D immersed interface method for fluid–solid interaction, Comput. Methods Appl. Mech. Engrg. (2007), doi:10.1016/j.cma.2007.06.012

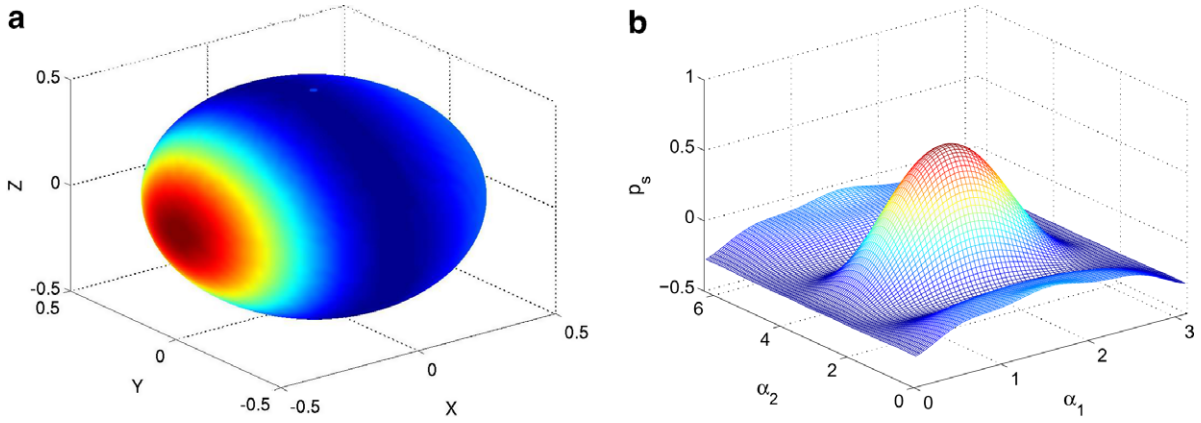
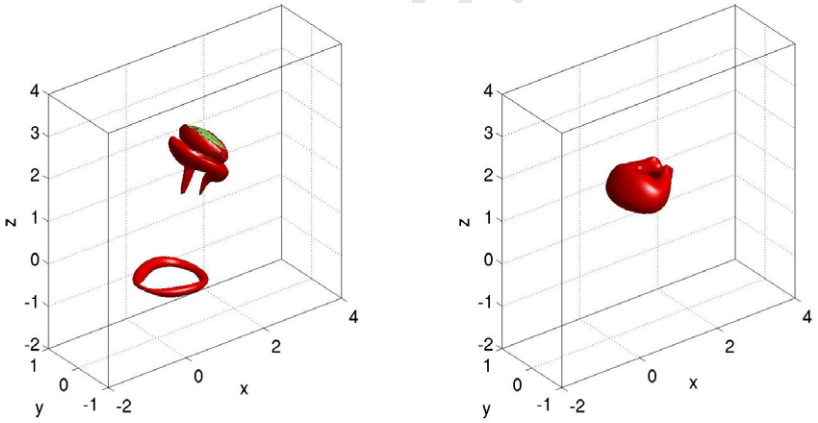


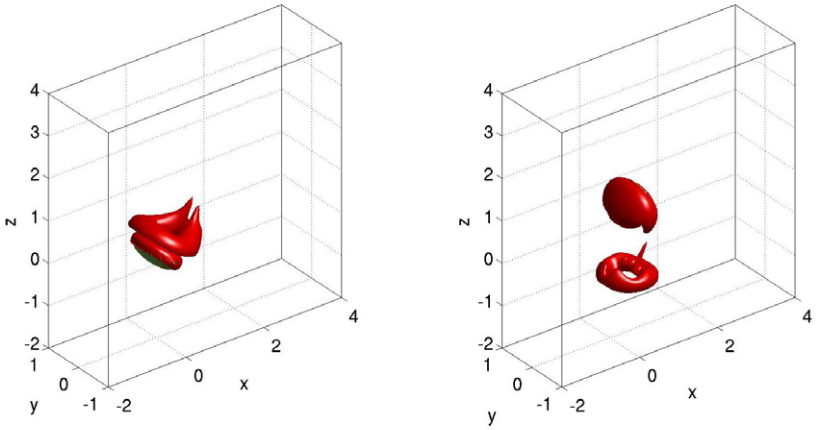
Fig. 22. Surface pressure distribution at $Re = 200$ in (a) Cartesian space, (b) parameter space.

where (x_c, y_c, z_c) are the coordinates of the flapper center, θ is the rotation angle of the flapper with respect to the x-axis, and $t_c = 1$ is a characteristic time of a ramping process to avoid the impulsive start of the flapper rotation. The rotation is around the flapper center, and the rotation direction is given by $\mathbf{R} = (0, -1, 0)$. The Reynolds number of the flow is $Re = 196$. The flapping period is $T_f = \frac{2\pi}{0.8}$. A 2D flapper with the similar kinematics has been simulated in [29,33].

Fig. 23 shows four snapshots of vortex structures during one flapping period. As suggested by Chong et al. [4], vortex structures can be identified by the isosurface of the unit value of Q, where Q is the second invariant of the rate-of-deformation tensor $\nabla \mathbf{v}$ and can be calculated by



(a) $t = 4.00T_f$ (b) $t = 4.25T_f$



(c) $t = 4.50T_f$ (d) $t = 4.75T_f$

Fig. 23. Snapshots of vortex structures shed from a flapper.

Please cite this article in press as: S. Xu, Z. Jane Wang, A 3D immersed interface method for fluid–solid interaction, Comput. Methods Appl. Mech. Engrg. (2007), doi:10.1016/j.cma.2007.06.012

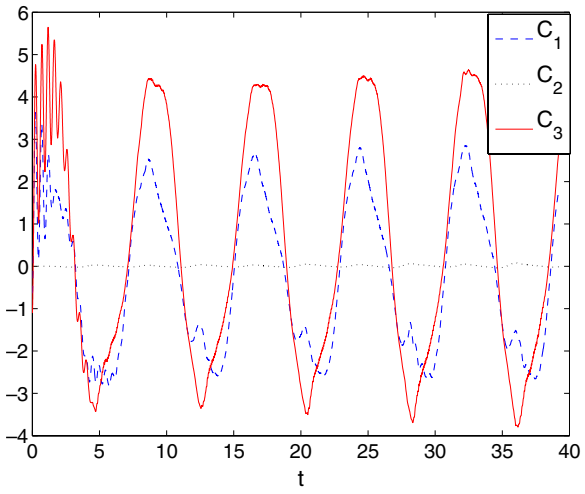


Fig. 24. Time history of force coefficients for flow around a flapper.

$Q = -\frac{1}{2}\nabla\mathbf{v} : \nabla\mathbf{v}$ for incompressible flow. Fig. 23 indicates that a vortex ring is shed downward in the upstroke of the flapper.

Fig. 24 shows the time history of the force coefficients C_1, C_2, and C_3. The force coefficients C_1, C_2, and C_3 are defined as $(C_1, C_2, C_3) = \mathbf{C} = 2\mathbf{G}/A$, where $A = \pi ab$ is the projected area of flat surfaces. In the current simulation setup, C_1 is the drag coefficient, C_3 is the lift coefficient, and C_2 is the side force coefficient. The time average values of C_1, C_2, and C_3 in the first 5 periods are -0.23, 0.0069, and 0.53, respectively.

7.5. Flow induced by multiple spinning spheres

In this last example, flow induced by different number of spinning spheres is simulated to examine the efficiency of the current method in handling multiple solids. Each sphere spins around an axis through its center, and the spinning direction is given by $\mathbf{R} = (0, 1, 0)$. Again, the model of singular force is given by Eq. (77) with $K_s = 1000$. Each sphere is represented by $M_1 \times M_2 = 32 \times 64$ Lagrangian markers. The simulation domain is shown in Fig. 25, and it is discretized with $N_x \times N_y \times N_z = 65 \times 33 \times 65$. Also shown in Fig. 25 is contours of u in the x–z plane for the case with four spheres.

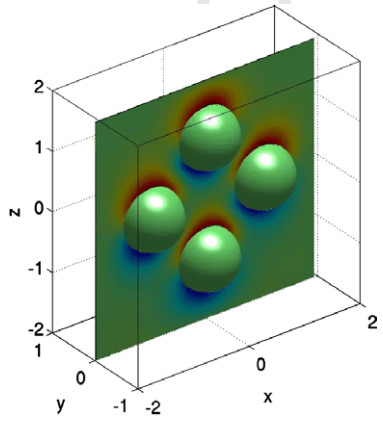


Fig. 25. Computational domain for flow involving multiple spinning spheres.

Table 6
Relative computational cost for different number of spheres

Number of objects	0	1	2	3	4
Relative computational cost	1	1.51	1.94	2.46	2.94

The computational time corresponding to the case with no sphere is 0.202 h.

Table 6 lists the relative computational cost spent on 5000 time steps using the same computer for different number of spheres, and indicates a linear relation with the slope equal to about 0.5. In this example, the ratio of $N_x \times N_y \times N_z$ to $M_1 \times M_2$ is about 65, which is relatively small. In the previous example of flow around a flapper, the ratio is about 4 times larger. If this ratio is larger, the slope is expected to become smaller, and the method is relatively more efficient in handling multiple moving solids.

8. Conclusions

This paper presents the detailed numerical implementation of a 3D immersed interface method with the jump conditions derived in [32]. The method is tested in simulating (a) flow inside a rotating object, (b) flow induced by a relaxing balloon, (c) flow past a stationary sphere, (d) flow around a flapper, and (d) flow induced by multiple spinning spheres. The test results suggest that (1) the method has near second-order accuracy in the infinity norm for velocity, and the accuracy for pressure is between first and second order; (2) the method conserves the volume enclosed by a no-penetration boundary very well; and (3) the method can efficiently handle multiple moving solids with ease. Future work on the method includes the analysis and improvement of its numerical stability, the use of different interface tracking schemes, and the application of the method to relatively high Reynolds numbers through implicit treatment of solid motion and adaptive mesh refinement.

Acknowledgements

S. Xu thanks Professor Charles Peskin's help with his career development. The authors want to thank Professor Randall LeVeque and Professor Zhilin Li for helpful discussions. This work is supported by the AFOSR.

Appendix. Analytical solutions to the linear systems in Section 3

The coefficient matrices C_1 and C_2 in the small linear systems given by Eqs. (28), (30), (33), and (37) are formed from three independent vectors, $\boldsymbol{\tau}$, $\mathbf{b}$, and $\mathbf{n}$. They are non-

singular (except at the two poles of a spherical interface). The inverse of C_1 is

$$C_1^{-1} = \frac{1}{J^2}\begin{pmatrix} t_1 & \beta_1 & n_1 \\ t_2 & \beta_2 & n_2 \\ t_3 & \beta_3 & n_3 \end{pmatrix}, \tag{94}$$

where $(t_1, t_2, t_3) := \mathbf{t} = \mathbf{b} \times \mathbf{n}$, and $(\beta_1, \beta_2, \beta_3) := \beta = \mathbf{n} \times \tau$.

The linear systems with the form of $C_2\mathbf{q} = \mathbf{r}$ can be expanded, split, and rewritten as below

$$\begin{pmatrix} C_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & C_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \tau_1 & \tau_2 & \tau_3 & 0 & 0 & 0 \\ 0 & \tau_1 & 0 & \tau_2 & \tau_3 & 0 \\ 0 & 0 & \tau_1 & 0 & \tau_2 & \tau_3 \\ b_1 & b_2 & b_3 & 0 & 0 & 0 \\ 0 & b_1 & 0 & b_2 & b_3 & 0 \\ 0 & 0 & b_1 & 0 & b_2 & b_3 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{pmatrix}$$
$$= \begin{pmatrix} r_1 \\ r_3 \\ r_4 \\ r_3 \\ r_2 \\ r_5 \\ r_6 \end{pmatrix} \Rightarrow \begin{pmatrix} \tau_1 & \tau_2 & \tau_3 & 0 & 0 & 0 \\ 0 & \tau_1 & 0 & \tau_2 & \tau_3 & 0 \\ 0 & 0 & \tau_1 & 0 & \tau_2 & \tau_3 \\ b_1 & b_2 & b_3 & 0 & 0 & 0 \\ 0 & b_1 & 0 & b_2 & b_3 & 0 \\ 0 & 0 & b_1 & 0 & b_2 & b_3 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{pmatrix}$$
$$= \begin{pmatrix} C_1^{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & C_1^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} r_1 \\ r_3 \\ r_4 \\ r_3 \\ r_2 \\ r_5 \\ r_6 \end{pmatrix} := (d_1, d_2, d_3, d_4, d_5, d_6, d_7)^{\mathrm{T}}, \tag{95}$$

where $\mathbf{q} = (q_1, q_2, q_3, q_4, q_5, q_6)^{\mathrm{T}}$ and $\mathbf{r} = (r_1, r_2, r_3, r_4, r_5, r_6)^{\mathrm{T}}$.

As $\boldsymbol{\tau}$ is non-zero, at least one component of $\boldsymbol{\tau}$ is non-zero. If $\tau_1 \neq 0$, then

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ \tau_1 & \tau_2 & \tau_3 & 0 & 0 & 0 \\ b_1 & b_2 & b_3 & 0 & 0 & 0 \\ 0 & \tau_1 & 0 & \tau_2 & \tau_3 & 0 \\ 0 & b_1 & 0 & b_2 & b_3 & 0 \\ 0 & 0 & \tau_1 & 0 & \tau_2 & \tau_3 \\ 0 & 0 & b_1 & 0 & b_2 & b_3 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{pmatrix} = \begin{pmatrix} d_7 \\ d_1 \\ d_4 \\ d_2 \\ d_5 \\ d_3 \\ d_6 \end{pmatrix}. \tag{96}$$

If $\tau_2 \neq 0$, then

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ \tau_2 & \tau_3 & \tau_1 & 0 & 0 & 0 \\ b_2 & b_3 & b_1 & 0 & 0 & 0 \\ 0 & \tau_2 & 0 & \tau_3 & \tau_1 & 0 \\ 0 & b_2 & 0 & b_3 & b_1 & 0 \\ 0 & 0 & \tau_2 & 0 & \tau_3 & \tau_1 \\ 0 & 0 & b_2 & 0 & b_3 & b_1 \end{pmatrix} \begin{pmatrix} q_4 \\ q_5 \\ q_2 \\ q_6 \\ q_3 \\ q_1 \end{pmatrix} = \begin{pmatrix} d_7 \\ d_2 \\ d_5 \\ d_3 \\ d_6 \\ d_1 \\ d_4 \end{pmatrix}. \tag{97}$$

If $\tau_3 \neq 0$, then

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ \tau_3 & \tau_2 & \tau_1 & 0 & 0 & 0 \\ b_3 & b_2 & b_1 & 0 & 0 & 0 \\ 0 & \tau_3 & 0 & \tau_2 & \tau_1 & 0 \\ 0 & b_3 & 0 & b_2 & b_1 & 0 \\ 0 & 0 & \tau_3 & 0 & \tau_2 & \tau_1 \\ 0 & 0 & b_3 & 0 & b_2 & b_1 \end{pmatrix} \begin{pmatrix} q_6 \\ q_5 \\ q_3 \\ q_4 \\ q_2 \\ q_1 \end{pmatrix} = \begin{pmatrix} d_7 \\ d_3 \\ d_6 \\ d_2 \\ d_5 \\ d_1 \\ d_4 \end{pmatrix}. \tag{98}$$

Eqs. (96)–(98) can be denoted with the following form:

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ a & b & c & 0 & 0 & 0 \\ x & y & z & 0 & 0 & 0 \\ 0 & a & 0 & b & c & 0 \\ 0 & x & 0 & y & z & 0 \\ 0 & 0 & a & 0 & b & c \\ 0 & 0 & x & 0 & y & z \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{pmatrix} = \begin{pmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \\ R_7 \end{pmatrix}, \tag{99}$$

where $a \neq 0$. With Gaussian elimination, the coefficient matrix in Eq. (99) can be transformed to

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & a & 0 & b & c & 0 \\ 0 & 0 & a & 0 & b & c \\ 0 & 0 & 0 & -s_3 & -2bc & -s_2 \\ 0 & 0 & 0 & 0 & e & f \\ 0 & 0 & 0 & 0 & g & h \\ 0 & 0 & 0 & 0 & m_3 & -m_2 \end{pmatrix}, \tag{100}$$

where

$$s_2 = a^2 + c^2 > 0,$$
$$s_3 = a^2 + b^2 > 0,$$
$$m_2 = cx - az,$$
$$m_3 = ay - bx,$$
$$e = 2bc(ax + by) - s_3(cy + bz),$$
$$f = s_2(ax + by) - s_3(ax + cz),$$
$$g = -s_3 m_2 - 2bcm_3,$$
$$h = -s_2 m_3.$$

Since a and $\mathbf{n}$ are non-zero except at the poles of an ellipsoidal interface, one of m_2 and m_3 must be non-zero [32]. The right lower corner matrix in (100) can be transformed to

$$\begin{pmatrix} 0 & m_3 f + m_2 e \\ 0 & m_3 h + m_2 g \\ m_3 & -m_2 \end{pmatrix}, \tag{101}$$

if $m_3 \neq 0$, or

$$\begin{pmatrix} m_3 f + m_2 e & 0 \\ m_3 h + m_2 g & 0 \\ m_3 & -m_2 \end{pmatrix}, \tag{102}$$

if $m_2 \neq 0$. It can be shown that $m_3 h + m_2 g < 0$ [32]. So far, Eq. (95) has been solved analytically.

References

[1] Attila J. Bergou, Sheng Xu, Z. Jane Wang, Passive wing rotation in dragonfly flight, J. Fluid Mech., submitted for publication.
[2] Petter Andreas Berthelsen, A decomposed immersed interface method for variable coefficient elliptic equations with non-smooth and discontinuous solutions, J. Comput. Phys. 197 (2004) 364–386.
[3] 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.
[4] M.S. Chong, A.E. Perry, B.J. Cantwell, A general classification of three-dimensional flow fields, Phys. Fluids A 2 (5) (1990) 765–777.
[5] R. Cortez, M. Minion, The blob projection method for immersed boundary problems, J. Comput. Phys. 161 (2000) 428–453.
[6] Shaozhong Deng, Kazufumi Ito, Zhilin Li, Three-dimensional elliptic solvers for interface problems and applications, J. Comput. Phys. 184 (2003) 215–243.
[7] E . Weinan, Jian-Guo Liu, Vorticity boundary condition and related issues for finite difference schemes, J. Comput. Phys. 124 (1996) 368–382.
[8] Aaron L. Fogelson, James P. Keener, Immersed interface method for Neumann and related problems in two and three dimensions, SIAM J. Sci. Comput. 22 (5) (2000) 1630–1654.
[9] Hans Johnston, Jian-Guo Liu, Finite difference schemes for incompressible flow based on local pressure boundary conditions, J. Comput. Phys. 180 (2002) 120–154.
[10] Hans Johnston, Jian-Guo Liu, Accurate stable and efficient Navier–Stokes solvers based on explicit treatment of the pressure term, J. Comput. Phys. 199 (2004) 221–259.
[11] T.A. Johnson, V.C. Patel, Flow past sphere up to a Reynolds number of 300, J. Fluid Mech. 378 (1999) 19–70.
[12] 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.
[13] 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.
[14] Long Lee, Randall J. LeVeque, An Immersed Interface Method for Incompressible Navier–Stokes Equations, SIAM J. Sci. Comput. 25 (3) (2003) 832–856.
[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 Frontiers in Applied Mathematics, vol. 33, 2006, ISBN: 0-89971-609-8.
[19] Mark N. Linnick, Hermann F. Fasel, A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains, J. Comput. Phys. 204 (2005) 157–192.
[20] Charles S. Peskin, Flow patterns around heart valves: a numerical method, J. Comput. Phys. 10 (1972) 252–271.
[21] Charles S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220–252.
[22] Charles S. Peskin, Beth Feller Printz, Improved volume conservation in the computation of flows with immersed elastic boundaries, J. Comput. Phys. 105 (1993) 33–46.
[23] Charles S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.
[24] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, Numerical Recipes in Fortran 77, second ed., The Art of Scientific Computing, 1999, pp. 848–852.
[25] A.M. Roma, C.S. Peskin, M.J. Berger, An adaptive version of the immersed boundary method, J. Comput. Phys. 153 (1999) 509–534.
[26] 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.
[27] 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.
[28] C. Tu, C.S. Peskin, Stability and instability in the computation of flows with moving immersed boundaries, SIAM J. Statist. Comput. 13 (1992) 70–83.
[29] Z. Jane Wang, Two dimensional mechanism for insect hovering, Phys. Rev. Lett. 85 (10) (2000).
[30] Andreas Wiegmann, Kenneth P. Bube, The immersed interface method for nonlinear differential equations with discontinuous coefficients and singular sources, SIAM J. Numer. Anal. 35 (1) (1998) 177–200.
[31] Andreas Wiegmann, Kenneth P. Bube, The explicit-jump immersed interface method: finite difference methods for PDEs with piecewise smooth solutions, SIAM J. Numer. Anal. 37 (3) (2000) 827–862.
[32] 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) 948–1980.
[33] 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.
[34] Luoding Zhu, Charles S. Peskin, Simulation of a flapping flexible filament in a flowing soap film by the immersed boundary method, J. Comput. Phys. 179 (2002) 452–468.

Please cite this article in press as: S. Xu, Z. Jane Wang, A 3D immersed interface method for fluid–solid interaction, Comput. Methods Appl. Mech. Engrg. (2007), doi:10.1016/j.cma.2007.06.012