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

\[
\frac{\partial \mathbf{v}}{\partial t} + \nabla \cdot (\mathbf{v} \mathbf{v}) = -\nabla p + \frac{1}{Re} \nabla^2 \mathbf{v} + \int_B \mathbf{f}(x_1, x_2, t) \delta(x - X(x_1, x_2, t)) \, dx_1 \, dx_2,
\]

(1)

\[
\nabla \cdot \mathbf{v} = 0,
\]

(2)

where \( \mathbf{v} \) is velocity, \( p \) is pressure, \( t \) is time, \( Re \) is the Reynolds number, \( B \) is the boundary of a solid, \( \delta(x - X(x_1, x_2, t)) \) is the 3D Dirac δ function, \( x \) is Cartesian coordinates, \( X(x_1, x_2, t) \) is the coordinates of the boundary,
and $f$ is the density of a singular force in the parameter space which is formed by two Lagrangian parameters $x_1$ and $x_2$ 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 $\delta$ 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]. Motivations for the presentation of this paper. If multiple boundary 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]:

$$ g(s_{m+1}) = \sum_{n=0}^{\infty} \frac{g^{(n)}(s_i)}{n!} (s_{m+1} - s_i)^n + \frac{1}{2} \sum_{n=0}^{\infty} \frac{g^{(n)}(\xi)}{n!} (s_{m+1} - s_i)^n + O(h^2), $$

where $g(s)$ is a non-smooth and discontinuous function as shown in Fig. 2a, and $[g^{(n)}(s_i)]$ denotes jump conditions along the $s$-axis, i.e., $[g^{(n)}(s_i)] = g^{(n)}(s_i) - g^{(n)}(s_{m+1})$. Second-order central finite difference schemes with discontinuities at $\xi$ and $\eta$ on its stencil shown in Fig. 2b can be modified as follows to keep their second-order accuracy:

$$ \frac{dg(s_i)}{ds} = g(s_{i+1}) - g(s_{i+1}) + \frac{1}{2} \sum_{n=0}^{\infty} \frac{g^{(n)}(\xi)}{n!} (s_{i+1} - s_i)^n + O(h^2), $$

$$ \frac{d^2g(s_i)}{ds^2} = g(s_{i+1}) - 2g(s_i) + g(s_{i+1}) + \frac{1}{2} \left( \sum_{n=0}^{\infty} \frac{g^{(n)}(\xi)}{n!} (s_{i+1} - s_i)^n \right) + O(h^2). $$

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:

$$ g(s_i) = \frac{g(s_{i+1}) + g(s_{i+1}) + O(h^2)}{2} + \frac{1}{2} \frac{dg(\xi)}{ds} (s_{i+1} - \xi) - \frac{1}{2} \frac{dg(\eta)}{ds} (s_{i+1} - \eta). $$

The jump conditions derived in [32] have been employed in an immersed interface method to simulate the interaction...
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 simula-
tion, and (3) conserves volumes enclosed by non-penetrat-
tion boundaries.

In this paper, the derived jump conditions are used in an
immersed interface method to simulate fluid–solid interac-
tion in 3D. Compared with the existing 3D immersed
boundary method, the current method has the improve-
ments 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 bound-
ary 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 $O(N \ln N) + O(M \ln M) + O(N) + O(M)$, 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 over-
view 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 triangula-
tion of an interface is introduced. The parametric triangula-
tion 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
\[
\begin{align*}
\nabla \cdot \left( \int_{S} f(x_1, x_2, t) \delta(x - X(x_1, x_2, t)) \, dx_1 \, dx_2 \right) & = 0.
\end{align*}
\]
where $D = \nabla \cdot \mathbf{v}$ is the divergence of the velocity, and $s_p$ is
\[
s_p = 2 \left( \frac{\partial u}{\partial x} \frac{\partial v}{\partial y} + \frac{\partial u}{\partial y} \frac{\partial v}{\partial x} + \frac{\partial u}{\partial z} \frac{\partial w}{\partial x} + \frac{\partial v}{\partial z} \frac{\partial w}{\partial y} + \frac{\partial w}{\partial x} \frac{\partial v}{\partial z} - \frac{\partial w}{\partial y} \frac{\partial v}{\partial z} \right). \tag{8}
\]
Terms with the divergence $D$ are kept in Eq. (7) to better
enforce the divergence-free condition, and $\frac{\partial u}{\partial x}$ 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
$\delta_x$, $\delta_y$, $\delta_z$, $\delta_{xx}$, $\delta_{yy}$, and $\delta_{zz}$ as
\[
\begin{align*}
\delta_x(i,j,k) &= \frac{(i+1/2,j,k) - (i-1/2,j,k)}{\Delta x} + c_x(i,j,k), \\
\delta_y(i,j,k) &= \frac{(i,j+1/2,k) - (i,j-1/2,k)}{\Delta y} + c_y(i,j,k), \\
\delta_z(i,j,k) &= \frac{(i,j,k+1/2) - (i,j,k-1/2)}{\Delta z} + c_z(i,j,k), \\
\delta_{xx}(i,j,k) &= \frac{(i+1,j,k) - 2(i,j,k) + (i-1,j,k)}{\Delta x^2} + c_{xx}(i,j,k), \\
\delta_{yy}(i,j,k) &= \frac{(i,j+1,k) - 2(i,j,k) + (i,j-1,k)}{\Delta y^2} + c_{yy}(i,j,k), \\
\delta_{zz}(i,j,k) &= \frac{(i,j,k+1) - 2(i,j,k) + (i,j,k-1)}{\Delta z^2} + c_{zz}(i,j,k),
\end{align*}
\]
where $\Delta x$, $\Delta y$, $\Delta z$ 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 recov-
ered. 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 $(i + \frac{1}{2}, j, k)$ can be written as

$$\frac{\partial u}{\partial t} = -\delta_i(\omega u) - \delta_j(\nu u) - \delta_k(u) - \bar{p} + \frac{1}{Re} (\delta_{xx} + \delta_{yy} + \delta_{zz})u,$$

(15)

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

$$\left(\delta_{xx} + \delta_{yy} + \delta_{zz}\right)p = -\frac{\partial D}{\partial t} - 2\delta_i(\nu uD) + \delta_j(\nu vD) + \delta_k(\nu wD) + \frac{1}{Re} (\delta_{xx} + \delta_{yy} + \delta_{zz})D + s_p,$$

(16)

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

$$s_p = 2(\delta_i\nu \partial_i u - \delta_\nu \partial_i v + \delta_j\nu \partial_j w - \delta_\nu \partial_j w + \delta_k\nu \partial_k w - \nu \partial_k w).$$

(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 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 $\Delta p$ 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].

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 $\tau$ and $b$, and the normal vector $n$ shown in Fig. 1 appear in the expressions for the jump conditions below. They are defined as follows:

$$\tau = (\tau_1, \tau_2, \tau_3) = \frac{\partial X}{\partial x_1} = \left(\frac{\partial X}{\partial x_1}, \frac{\partial X}{\partial x_2}, \frac{\partial X}{\partial x_3}\right),$$

(18)

$$b = (b_1, b_2, b_3) = \frac{\partial X}{\partial x_2} = \left(\frac{\partial X}{\partial x_1}, \frac{\partial X}{\partial x_2}, \frac{\partial X}{\partial x_3}\right),$$

(19)

$$n = (n_1, n_2, n_3) = \tau \times b.$$  

(20)

The parameters $x_1$ and $x_2$ are chosen such that the vector $n$ points to outside a solid. In addition, the following definitions are used:

$$J = ||n||,$$

(21)

$$n^* = n,$$

(22)

$$F = F_n n^*,$$

(23)

$$F_n = F \cdot n^*,$$

(24)

$$F_t = F - F_n n^*,$$

(25)

where $n^*$ is the unit normal vector, and $F$ is the density of singular force in the Cartesian space. In the numerical examples presented in Section 7, the force density $F$ is calculated based on force models that relate the force density $F$ to the configuration of the Lagrangian markers. The
The jump conditions for the first derivatives of the velocity are

\[ \frac{\partial \mathbf{v}_1}{\partial n} = 0, \quad \frac{\partial \mathbf{v}_2}{\partial n} = 0, \quad \frac{\partial \mathbf{v}_3}{\partial n} = \frac{-Re \mathbf{F}_2}{\mathbf{F}}. \]  

(28)

where \( \frac{\partial}{\partial n} \) denotes jump conditions along the direction of \( n \), and the coefficient matrix \( C_1 \) is

\[ C_1 = \begin{pmatrix} \tau_1 & \tau_2 & \tau_3 \\ b_1 & b_2 & b_3 \\ n_1 & n_2 & n_3 \end{pmatrix}. \]  

(29)

The jump conditions for the first derivatives of the pressure satisfy

\[ C_1 \begin{pmatrix} \frac{\partial p_1}{\partial x} \\ \frac{\partial p_1}{\partial y} \\ \frac{\partial p_1}{\partial z} \end{pmatrix} = \begin{pmatrix} \frac{\partial p_2}{\partial x} \\ \frac{\partial p_2}{\partial y} \\ \frac{\partial p_2}{\partial z} \end{pmatrix} \]  

\[ \text{and} \quad \frac{\partial p_1}{\partial x} = \left( \frac{\partial p_1}{\partial x_2} \right) + \left( \frac{\partial p_2}{\partial x_2} \right), \]  

(30)

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

\[ \tilde{f}_1 = (\mathbf{b} \times \mathbf{n}) \cdot \mathbf{F}, \]  

\[ \tilde{f}_2 = (\mathbf{n} \times \mathbf{\tau}) \cdot \mathbf{F}. \]  

(31)

The jump conditions for the second derivatives of the velocity satisfy

\[ C_2 = \begin{pmatrix} \tau_{11} & \tau_{12} + \tau_{31} & \tau_{13} + \tau_{31} \\ b_{11} & b_{12} + b_{31} & b_{13} + b_{31} \\ n_{11} & n_{12} + n_{31} & n_{13} + n_{31} \end{pmatrix} \]  

\[ \begin{pmatrix} \tau_{21} & \tau_{22} & \tau_{23} \\ b_{21} & b_{22} & b_{23} \\ n_{21} & n_{22} & n_{23} \end{pmatrix} \]  

\[ \begin{pmatrix} \tau_{31} & \tau_{32} & \tau_{33} \\ b_{31} & b_{32} & b_{33} \\ n_{31} & n_{32} & n_{33} \end{pmatrix} \]  

\[ C_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -Re \frac{\partial p_1}{\partial x_2} \end{pmatrix} \]  

\[ \text{where} \quad C_2 = \begin{pmatrix} \frac{\partial^2 \mathbf{F}_1}{\partial x_2^2} \frac{\partial \mathbf{F}_1}{\partial x_1} \\ \frac{\partial^2 \mathbf{F}_1}{\partial y_2^2} \frac{\partial \mathbf{F}_1}{\partial x_1} \\ \frac{\partial^2 \mathbf{F}_1}{\partial z_2^2} \frac{\partial \mathbf{F}_1}{\partial x_1} \\ \frac{\partial^2 \mathbf{F}_1}{\partial x_2^2} \frac{\partial \mathbf{F}_1}{\partial y_1} \\ \frac{\partial^2 \mathbf{F}_1}{\partial y_2^2} \frac{\partial \mathbf{F}_1}{\partial y_1} \\ \frac{\partial^2 \mathbf{F}_1}{\partial z_2^2} \frac{\partial \mathbf{F}_1}{\partial y_1} \\ \frac{\partial^2 \mathbf{F}_1}{\partial x_2^2} \frac{\partial \mathbf{F}_1}{\partial z_1} \\ \frac{\partial^2 \mathbf{F}_1}{\partial y_2^2} \frac{\partial \mathbf{F}_1}{\partial z_1} \\ \frac{\partial^2 \mathbf{F}_1}{\partial z_2^2} \frac{\partial \mathbf{F}_1}{\partial z_1} \end{pmatrix}. \]  

(33)

The calculation of surface derivatives with respect to \( x_1 \) and \( x_2 \) 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 .

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
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 \(x_1 \in [0, \pi]\) and \(x_2 \in [0, 2\pi]\). 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:

\[
x_{1m_1} = \frac{\Delta x_1}{2} + m_1 \Delta x_1, \quad m_1 = 0, 1, \ldots, M_1,
\]

\[
x_{2m_2} = m_2 \Delta x_2, \quad m_2 = 0, 1, \ldots, M_2,
\]

where \(\Delta x_1 = \frac{s_0}{M_1+1}, \Delta x_2 = \frac{2\pi}{M_2}\), 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 \(x_1 = 0\) and \(x_1 = \pi\). 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, \(\mathbf{F} = \frac{1}{\rho}\), 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 \(x_2\), The first derivatives with respect to \(x_2\) in Section 3 are calculated numerically using periodic cubic splines. The second derivatives with respect to \(x_2\) 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 \(x_1\), a closed smooth curve on the interface for each \(x_{2m_2} \in [0, \pi]\) is composed by two branches corresponding to values of \(x_{2m_1}\) and \(\pi + x_{2m_2}\), as demonstrated in Fig. 5a. So \(M_2\) has to be even. It can be shown that the functions \(\mathbf{F}_1, f_1\), and \(\mathbf{F}_2, f_2\) 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 \(x_1\) in Section 3 are smooth on the curve. On the branch corresponding to \(x_2 = x_{2m_1}\),

\[
x_1^* = x_1^1,
\]

\[
\frac{\partial}{\partial x_1^*} = \frac{\partial}{\partial x_1^{1n}},
\]

where \(x_1^1\) is defined in Fig. 5a, and \(n = 0, 1, 2, \ldots\) On the branch corresponding to \(x_2 = \pi + x_{2m_2}\),

\[
x_1^* = 2\pi - x_1^1,
\]

\[
\frac{\partial}{\partial x_1^*} = \frac{\partial}{\partial x_1^{2n}},
\]

\[
\frac{\partial}{\partial x_1^{2n+1}} = - \frac{\partial}{\partial x_1^{2n+1}},
\]

Thus, periodic cubic splines with respect to \(x_1^*\) can be used to calculate \(\frac{\partial}{\partial x_1}\) with the above transformations. The cost count to compute all the surface derivatives in Section 3 is \(O(M)\).
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 \( x_2 \), original data defined on the parameter regions "o" and "p" in Fig. 5b are directly used because of their periodicity in \( x_2 \). To filter in the direction of \( x_1 \), 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 \( x_1 \) start at \( \frac{2\pi}{M_1} \) and end at \( \pi - \frac{2\pi}{M_1} \). 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

\[
X_s = r \cos(z_1), \\
Y_s = (R + r \sin(z_1)) \cos(z_2), \\
Z_s = (R + r \sin(z_1)) \sin(z_2),
\]

where \( R \) is the distance from the center of the torus tube to the center of the torus, and \( r \) is the radius of the tube, and \( z_1 \in [0, 2\pi] \) and \( z_2 \in [0, 2\pi] \). A Lagrangian interface mesh as illustrated in Fig. 4b is generated with the following parameter discretization:

\[
x_{1m1} = m_1 \Delta x_1, \quad m_1 = 0, 1, \ldots, M_1, \\
x_{2m2} = m_2 \Delta x_2, \quad m_2 = 0, 1, \ldots, M_2,
\]

where \( \Delta x_1 = \frac{2\pi}{M_1} \) and \( \Delta x_2 = \frac{2\pi}{M_2} \). 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 \( x_1 \) and \( x_2 \). 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 be are 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):

\[
\Delta P_1(x_{1m1}, x_{2m1}), \quad P_2(x_{1m1} + \Delta x_1, x_{2m1}), \quad P_3(x_{1m1}, x_{2m1} + \Delta x_2), \quad \text{and} \quad \Delta P_4(x_{1m1}, x_{2m1} + \Delta x_2).
\]

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 \( z_1 = 0 \) in Fig. 7.

The interfaces between an interface and MAC grid lines are found by projecting triangular patches along the \( x_-, y-, \) and \( z- \) axes. 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 \( I I I I I I \). The rectangle is used to determine MAC grid lines which are parallel to the \( z- \) axis and may intersect \( \Delta P_1P_2P_4 \). If the projection \( Q_3(x_F, y_J) \) 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 \( (x_F, y_J) \). 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:

\[
g(z_1, z_2) = c_{g0} + c_{g1}z_1 + c_{g2}z_2,
\]

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:

\[
g(z_{1m1}, z_{2m1}) = c_{g0} + c_{g1}z_{1m1} + c_{g2}z_{2m1}, \quad g(z_{1m1} + \Delta x_1, z_{2m1}) = c_{g0} + c_{g1}(z_{1m1} + \Delta x_1) + c_{g2}z_{2m1}, \quad g(z_{1m1}, z_{2m1} + \Delta x_2) = c_{g0} + c_{g1}z_{1m1} + c_{g2}(z_{2m1} + \Delta x_2).
\]
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:

\[
(y - Y_1)(X_2 - X_1) = (x - X_1)(Y_2 - Y_1),
\]

\[
(y - Y_2)(X_1 - X_2) = (x - X_2)(Y_1 - Y_2),
\]

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

---

![Fig. 8. Projection of a triangular patch for finding intersections between an interface and grid lines.](image)

![Fig. 9. Intersection on an edge: (a) recorded, (b) discarded.](image)

![Fig. 10. Intersection on a vertex: (a) recorded, (b) discarded.](image)
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 \( u_{i,j+k}, u_{i,j+k'}, v_{i,j+k}, v_{i,j+k'}, w_{i,j+k}, w_{i,j+k'} \) and \( \theta_{i,j+k}, \theta_{i,j+k'}, \theta_{i,j+k}, \theta_{i,j+k'} \).

Define the interpolation operators \( e_i, e_j, e_k \) as

\[
\begin{align*}
\tilde{u}_i(j,k) &= \frac{u_{i,j+k} + u_{i,j-k}}{2} + c_i u_{i,j,k}, \\
\tilde{v}_j(j,k) &= \frac{v_{i,j+k} + v_{i,j-k}}{2} + c_j v_{i,j,k}, \\
\tilde{w}_k(j,k) &= \frac{w_{i,j+k} + w_{i,j-k}}{2} + c_k w_{i,j,k},
\end{align*}
\]

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:

\[
\begin{align*}
\tilde{u}_{i,j,k} &= e_i u_{i,j,k}, \\
\tilde{u}_{i,j+k} &= e_i u_{i,j+k}, \\
\tilde{u}_{i,j+k/2} &= e_i u_{i,j+k/2},
\end{align*}
\]

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 \( n^* \) at the marker, and the two supplemental points are at the different sides of the interface with the equal distance \( \Delta n^* \) away from the interface. According to Eq. (6), the interpolation scheme at this step is

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

where the normal derivative \( \left( \frac{\partial n}{\partial s} \right) = -Re \mathbf{F} \), 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 \( \Delta n^* \) 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:

\[
\begin{align*}
\tilde{u}(II) &= \frac{z(III) - z(III)}{z(III) - z(I)} u(I) + \frac{z(II) - z(I)}{z(III) - z(I)} u(II),
\end{align*}
\]

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 lines and fluid–solid interfaces is of order \( M \) too.)

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

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 \( \Omega \) 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

\[
\mathbf{v}(x) = \Omega \mathbf{R} \times x, \quad (75)
\]

\[
p(x) = \frac{1}{2} \Omega^2 \| \mathbf{R} \times x \|^2 + p_0, \quad (76)
\]

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:

\[
\mathbf{F} = K_s (x_c - x), \quad (77)
\]

where \( K_s \) is the spring stiffness, and \( x_c \) 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 \( l \) at \( Re = 10 \). The angular velocity of the rotation is \( \Omega = 1 \), and the rotation direction is given by \( \mathbf{R} = (\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}) \). 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:

\[
\text{order} = \frac{\ln(\| e_{\text{current}} \|_\infty / \| e_{\text{previous}} \|_\infty )}{\ln(A_{\text{current}} / A_{\text{previous}})}, \quad (78)
\]

where \( e_{\text{current}} \) and \( e_{\text{previous}} \) denote the errors at the current and the previous rows in Table 2, respectively, and \( A_{\text{current}} \) and \( A_{\text{previous}} \) 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...
of spring stiffness. The spatial resolution of the simulation corresponds to $N_x \times N_y \times N_z = 33 \times 33 \times 33$ and $M_1 \times M_2 = 32 \times 64$.

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 $K_e = 10$ and $K_e = 5000$ 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 equilibrium shape from the initial distortion. The initial velocity and the pressure are set zero, and the coupled equilibrium pressure at two slices shown in Fig. 15 indicates that it is piecewise constants with a jump, $p = -\frac{\rho}{\rho} \frac{\nabla H}{c}$. The density of singular force is modeled by

$$F = EH \nu^n,$$

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, $p = -\frac{\rho}{\rho} \frac{\nabla H}{c}$. The density of singular force is modeled by

$$F = EH \nu^n.$$
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 $CFL_l$ are defined as

$$CFL_c = \frac{\Delta t}{\Delta x} \left( \frac{u_{\text{max}}}{\Delta x} + \frac{v_{\text{max}}}{\Delta y} + \frac{w_{\text{max}}}{\Delta z} \right),$$

$$CFL_l = \frac{\Delta t}{\Delta x} \left( \frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2} \right),$$

where $u_{\text{max}}$, $v_{\text{max}}$, and $w_{\text{max}}$ 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, $Re = 10, 20, 100, \text{ and } 200$, is simulated in this example. The spring model given by Eq. (77) is used to calculate the density of singular force with $K_s = 1000, 5000, 2000, \text{ and } 100$ for $Re = 10, 20, 100, \text{ and } 200$, 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 $Re = 10, 20, \text{ and } 100$, the domain sizes are $[-4.16, -4.16, -4.16]$ 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 $[-2.8, -2.8, -2.8]$, and the time step is controlled by $CFL_c = CFL_l = 0.2$. 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:

$$\frac{\partial p}{\partial x} = 1 \frac{\partial^2 u}{\partial x^2},$$

At $Re = 10, 20, \text{ and } 100$, the initial flow field is set uniform with $u = 1$ as the far field, and the sphere impulsively stops
is used to ramp the far-field uniform flow from \( u = 0 \) to \( u = 1 \) to avoid impulsive stop of the sphere

\[
\begin{align*}
    u_n = u_a + \exp \left( -\frac{t_n - t_0}{t_e} \right) - \exp \left( -\frac{t_n}{t_e} \right), \\
    n = 1, 2, \ldots,
\end{align*}
\]

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

\[
\mathbf{G} = \int_S \left( -p^+ \mathbf{n}^+ + \frac{1}{Re} \left( \frac{\partial \mathbf{v}}{\partial \mathbf{n}} \right)^+ \right) dS
\]

\[
= -\int_S \textbf{f} \, dx_1 \, dx_2 + \int_S \left( -p^+ \mathbf{n}^+ + \frac{1}{Re} \left( \frac{\partial \mathbf{v}}{\partial \mathbf{n}} \right) \right) dS,
\]

from the uniform velocity. At \( Re = 200 \), the initial flow field is set quiescent with \( u = 0 \), and the following formula

\[ u_n = \begin{cases} 
0 & \text{for } n = 0 \\
\frac{1}{C_0} \exp \left( -\frac{t_n}{C_0} \right) & \text{for } n = 1, 2, \ldots
\end{cases}
\]

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.

Fig. 15. Equilibrium pressure \( p \) at the two slices \( x = 0 \) and \( y = 0 \).

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.

where the subscript ‘+’ denotes the outer side of the object surface (fluid–solid interface) and the subscript ‘−’ denotes the inner side. The term \( \int_S f \, dx_1 \, dx_2 \) 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:

\[
\int_S (-p \cdot n) \, dS = V \frac{dU_t}{dt},
\]

(86)

\[
\int_S \left( \frac{\partial \mathbf{v}}{\partial t} \right) \, dS = 0,
\]

(87)

where \( U_t \) is the translational velocity of the object, and \( V \) is the object volume. Thus the calculation of the fluid force can be simplified to

\[
G = - \int_S f \, dx_1 \, dx_2 + V \frac{dU_t}{dt},
\]

(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

\[
G = - \int_S f \, dx_1 \, dx_2.
\]

(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 \( (C_1, C_2, C_3) = C = 2G/A \), where \( A = \frac{4}{3} \) 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.

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 \( Re = 10 \).
The separation angle $\Theta_s$, 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 $p_s = F_n$ 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 $\frac{a}{2} \times \frac{b}{2} \times \frac{c}{2}$ and $M_1 \times M_2 = 32 \times 64$. The time step is fixed in this simulation, and it is $\Delta t = 1.96 \times 10^{-3} \approx \frac{T_f}{400}$, where $T_f$ is the flapping period.

The motion of the flapper is formulated as

\[ x_c = 1.25 (\cos(0.8t) + 1) \cos \left( \frac{\pi}{3} \right), \]  
\[ y_c = 0, \]  
\[ z_c = 1.25 (\cos(0.8t) + 1) \sin \left( \frac{\pi}{3} \right), \]  
\[ \theta = \frac{3\pi}{4} + \frac{\pi}{4} \sin(0.8t) \left( 1 - \exp \left( -\frac{t}{t_c} \right) \right), \]

where $t_c$ is the flapping period.
where \((x_c, y_c, z_c)\) are the coordinates of the flapper center, \(\theta\) 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 \(R = (0, -1, 0)\). The Reynolds number of the flow is \(Re = 196\). The flapping period is \(T_f = \frac{2\pi}{\omega}\) 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

\[
Q = \frac{1}{2} \left( \mathbf{D}^2 \right) - \frac{1}{3} \left( \nabla \cdot \mathbf{v} \right)^2
\]

\[
\mathbf{D} = \nabla \mathbf{v} + (\nabla \mathbf{v})^T
\]

\[
\nabla \mathbf{v} = \begin{bmatrix} \partial v_x / \partial x \\ \partial v_y / \partial y \\ \partial v_z / \partial z \end{bmatrix}
\]

\[
\mathbf{D}^2 = \nabla \mathbf{v} \cdot (\nabla \mathbf{v})^T
\]

\[
\left( \nabla \cdot \mathbf{v} \right)^2 = (\partial v_x / \partial x + \partial v_y / \partial y + \partial v_z / \partial z)^2
\]

\[
\mathbf{v} = (v_x, v_y, v_z)
\]
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, $\mathbf{r}$, $\mathbf{b}$, and $\mathbf{n}$. They are non-

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

Fig. 25. Computational domain for flow involving multiple spinning spheres.
The inverse of $C_1$ is

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

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

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

$$\begin{pmatrix} C_1 & 0 & 0 \\ 0 & C_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_1 \\ r_3 \\ r_4 \end{pmatrix} = \begin{pmatrix} \tau_1 & \tau_2 & \tau_3 & 0 & 0 & 0 \\ 0 & \tau_1 & \tau_2 & \tau_3 & 0 & 0 \\ 0 & 0 & \tau_1 & \tau_2 & \tau_3 & 0 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix}. \quad (95)$$

If $\tau_3 \neq 0$, then

$$\begin{pmatrix} C_1^{-1} & 0 & 0 \\ 0 & C_1^{-1} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_1 \\ r_3 \\ r_4 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_3 \\ r_4 \end{pmatrix} = \begin{pmatrix} (d_1, d_2, d_3, d_4, d_5, d_6, d_7)^T, \end{pmatrix} \quad (96)$$

where $\mathbf{q} = (q_1, q_2, q_3, q_4, q_5, q_6, q_7)^T$ and $\mathbf{r} = (r_1, r_2, r_3, r_4, r_5, r_6, r_7)^T$.

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

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}. \quad (97)$$

If $\tau_2 \neq 0$, then

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & \tau_2 & \tau_3 & 0 & 0 & 0 \\ 0 & 0 & \tau_1 & \tau_2 & \tau_3 & 0 \\ 0 & b_1 & b_2 & b_3 & 0 & 0 \end{pmatrix} \begin{pmatrix} \tau_1 \\ \tau_3 \\ \tau_1 \\ b_1 \end{pmatrix} = \begin{pmatrix} \tau_2 \\ \tau_3 \\ \tau_1 \\ b_2 \end{pmatrix}. \quad (98)$$

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

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 \\ a & b & c & 0 & 0 & 0 \\ x & y & z & 0 & 0 & 0 \\ 0 & a & 0 & b & c & 0 \\ 0 & 0 & a & 0 & b & c \\ 0 & 0 & 0 & e & f & 0 \\ 0 & 0 & 0 & g & h & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{pmatrix} = \begin{pmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \end{pmatrix}, \quad (99)$$

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

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & a & 0 & b & c & 0 \\ 0 & 0 & a & 0 & b & c \\ 0 & 0 & 0 & s_3 & -2bc & -s_2 \\ 0 & 0 & 0 & e & f & 0 \\ 0 & 0 & 0 & g & h & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{pmatrix}, \quad (100)$$

where

$$\begin{align*}
s_2 &= a^2 + c^2 > 0, \\
s_3 &= a^2 + b^2 > 0, \\
m_2 &= ax - az, \\
m_3 &= ay - bx, \\
e &= 2bc(ax + by) - s_3(cy + bz), \\
f &= s_2(ax + by) - s_3(ax + cz), \\
g &= -s_2m_2 - 2bcm_3, \\
h &= -s_2m_3. \end{align*}$$

Since $a$ and $n$ 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
if \( m_1 \neq 0 \), or
\[
\begin{pmatrix}
m_3 f + m_2 e \\
0 \\
m_3 h + m_2 g \\
0
\end{pmatrix}
\]
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