INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids (2011)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/fld.2589

Simulation of three-dimensional flows over moving objects by an improved immersed boundary–lattice Boltzmann method

J. Wu [1,2] and C. Shu [2,*,†]

[1] *Department of Aerodynamics, College of Aerospace Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, China*
[2] *Department of Mechanical Engineering, National University of Singapore, Singapore*

SUMMARY

An improved immersed boundary–lattice Boltzmann method (IB–LBM) developed recently [28] was applied in this work to simulate three-dimensional (3D) flows over moving objects. By enforcing the non-slip boundary condition, the method could avoid any flow penetration to the wall. In the developed IB–LBM solver, the flow field is obtained on the non-uniform mesh by the efficient LBM that is based on the second-order one-dimensional interpolation. As a consequence, its coefficients could be computed simply. By simulating flows over a stationary sphere and torus [28] accurately and efficiently, the proposed IB–LBM showed its ability to handle 3D flow problems with curved boundaries. In this paper, we further applied this method to simulate 3D flows around moving boundaries. As a first example, the flow over a rotating sphere was simulated. The obtained results agreed very well with the previous data in the literature. Then, simulation of flow over a rotating torus was conducted. The capability of the improved IB–LBM for solving 3D flows over moving objects with complex geometries was demonstrated via the simulations of fish swimming and dragonfly flight. The numerical results displayed quantitative and qualitative agreement with the date in the literature. Copyright © 2011 John Wiley & Sons, Ltd.

Received 10 November 2010; Revised 24 February 2011; Accepted 26 March 2011

KEY WORDS: moving boundary; lattice Boltzmann method; immersed boundary method; three-dimensional flow; incompressible flow; numerical simulation

1. INTRODUCTION

As a long-term challenge in the computational fluid dynamics, the growing attention and interest have recently been put to the study of flows over moving objects both in applications and methodologies. Good examples include ball impacting on the floor, single/multiple particle sedimentation, insect flight and fish swimming, and so on. To accurately and efficiently simulate moving object problems is still at the frontier in the development of numerical techniques. By using structured or unstructured body-fitted grids, the traditional methods have been well developed and are commonly employed to simulate flows involving moving boundaries [1–3]. However, because of time-dependent mesh transformation or constant mesh regeneration process, these methods require high computational cost and may also introduce additional numerical errors. To develop more efficient numerical methods for moving boundary problems, it is desirable to decouple the solution of governing equations from the boundary. To make the solver as simple as possible, the fixed Cartesian mesh is usually adopted to discretize the governing equations. The effect of boundary on the flow field is often considered by two different ways. One is called the sharp interface approach, whereas the other is termed as the diffuse interface approach.

*Correspondence to: C. Shu, Department of Mechanical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore, 119260.
†E-mail: mpeshuc@nus.edu.sg

For the sharp interface approach, one popular example is the immersed interface method [4–9]. Its basic idea is to explicitly introduce jump conditions for the pressure and velocity across the interface. It should be indicated that the implementation of jump conditions is not an easy task. Another example in this family is the cut-cell method [10–13]. In this method, the fixed Cartesian cells are cut by the boundary surface that goes through the cells. Many irregular shapes of cells may appear in this method. One has to record all configurations of irregular shapes to correctly implement the boundary conditions. For moving boundary problems, this would greatly bring complexity into the programming.

For the diffuse interface approach, the most well-known example is the immersed boundary method (IBM). It was firstly proposed by Peskin [14] to study the cardiac mechanics and associated blood flows. In IBM, the boundary is represented by a set of Lagrangian points, and its effect on the flow field is depicted by the restoring force. The modified Navier–Stokes (N–S) equations with the force density, discretized on the fixed Cartesian (Eulerian) mesh, are then solved on the whole domain including the exterior and interior of the object. Following the work of Peskin [14], a number of efforts [15–17] have been devoted to further develop IBM. One effort is to use the lattice Boltzmann method (LBM) [18] instead of solving N–S equations to obtain the flow field. As compared to traditional N–S solvers, the most attractive features of LBM are its explicit operation and easy implementation. The pioneer work of combining IBM with LBM was made by Feng and Michaelides [19]. After that, a few variants have been proposed [20–24].

It is known that the accuracy of numerical simulation for flows over moving objects greatly depends on the computation of hydrodynamic forces exerted on the objects. In conventional IB–LBM [19–24], because the restoring force is pre-calculated, the non-slip boundary condition cannot be strictly satisfied. Consequently, the flow penetration to immersed boundary, which unavoidably induces momentum exchange across the boundary, will have occurred. Because the numerical force would be produced as a result of momentum exchange, the accuracy of force calculation on the objects would be affected. In this sense, satisfaction of the non-slip boundary condition in moving object simulation is very important. Recently, Wu and Shu [25] developed an improved version of IB–LBM. In this method, the restoring force is set as unknown and is solved by enforcing the non-slip boundary condition. As a result, no flow penetration is observed because of strict satisfaction of boundary conditions, and accurate force calculation is achieved. The improved IB–LBM has been successfully applied to simulate two-dimensional moving boundary flows [26] and particulate flows [27]. The numerical results obtained show good agreement with data in the literature. On the other hand, to improve the computational efficiency of LBM for a three-dimensional (3D) flow simulation, a new version of LBM was recently developed on the non-uniform Cartesian mesh [28], which is based on the second-order one-dimensional interpolation along straight lines. As compared with conventional versions of LBM such as Taylor series expansion and least squares-based LBM [29], the new approach requires much less interpolation coefficients to be computed. Hence, its computational efficiency is greatly enhanced. In this study, we will combine this new version of LBM [28] with the improved IB–LBM to simulate 3D flows over moving objects. As a validation, the simulation of laminar flow over a rotating sphere is carried out first. It is shown that numerical results are compared well with those from previous studies. Then, the flow over a rotating torus is simulated. Furthermore, to demonstrate the capability of the present solver for handling 3D flows with complex moving objects, the simulations of fish swimming and dragonfly flight are performed. The obtained results are quantitatively and qualitatively agreed with the data and findings in the literature.

2. AN IMPROVED IMMERSED BOUNDARY–LATTICE BOLTZMANN METHOD FOR THREE-DIMENSIONAL SIMULATION

For viscous incompressible flows with immersed boundaries, the N–S equations with external force density are generally used in IBM. Alternatively, we can also use LBM to solve the flow field. To efficiently and accurately simulate flows over objects in 3D space, an improved IB–LBM has been proposed in [28]. This method will be briefly described in this section.

2.1. *Three-dimensional boundary condition-enforced immersed boundary–lattice Boltzmann method*

In IB–LBM, the governing equations for 3D flows can be expressed as:

$$\mathrm{f}_\alpha(\mathbf{x}+\mathbf{e}_\alpha\delta t, t+\delta t)-\mathrm{f}_\alpha(\mathbf{x},t)=-\frac{1}{\tau}\left(\mathrm{f}_\alpha(\mathbf{x},t)-\mathrm{f}_\alpha^{\mathrm{eq}}(\mathbf{x},t)\right)+F_\alpha\delta t \tag{1}$$

$$F_\alpha=\left(1-\frac{1}{2\tau}\right)w_\alpha\left(\frac{\mathbf{e}_\alpha-\mathbf{u}}{c_s^2}+\frac{\mathbf{e}_\alpha\cdot\mathbf{u}}{c_s^4}\mathbf{e}_\alpha\right)\cdot\mathbf{f} \tag{2}$$

$$\rho\mathbf{u}=\sum_\alpha\mathbf{e}_\alpha\mathrm{f}_\alpha+\frac{1}{2}\mathbf{f}\delta t \tag{3}$$

where f_α is the distribution function, $\mathrm{f}_\alpha^{\mathrm{eq}}$ is its corresponding equilibrium state, τ is the single relaxation parameter, $\mathbf{e}_\alpha$ is the lattice velocity, $\mathbf{f}$ is the force density that is distributed from the boundary force, and w_α are the coefficients in the equilibrium distribution function. They depend on the selected lattice velocity model. In the current simulation, D3Q15 model is used, and the velocity set is given by:

$$\mathbf{e}_\alpha=\begin{cases}(0,0,0) & \alpha=0\\ (\pm1,0,0),(0,\pm1,0),(0,0,\pm1) & \alpha=1\sim6\\ (\pm1,\pm1,\pm1) & \alpha=7\sim14\end{cases} \tag{4}$$

The corresponding equilibrium distribution function is:

$$\mathrm{f}_\alpha^{\mathrm{eq}}(\mathbf{x},t)=\rho w_\alpha\left[1+\frac{\mathbf{e}_\alpha\cdot\mathbf{u}}{c_s^2}+\frac{(\mathbf{e}_\alpha\cdot\mathbf{u})^2-(c_s|\mathbf{u}|)^2}{2c_s^4}\right] \tag{5}$$

where $c_s^2=1/3$, $w_0=2/9$, $w_\alpha=1/9$ for $\alpha=1\sim6$, and $w_\alpha=1/72$ for $\alpha=7\sim14$. The key issue of IB–LBM is to calculate the force density $\mathbf{f}$ in Equations (2) and (3). As revealed in [28], if we define the intermediate fluid velocity $\mathbf{u}^*=\sum_\alpha\mathbf{e}_\alpha\mathrm{f}_\alpha/\rho$ and the fluid velocity correction $\delta\mathbf{u}=\mathbf{f}\delta t/2\rho$, Equation (3) can be re-expressed as:

$$\mathbf{u}=\mathbf{u}^*+\delta\mathbf{u} \tag{6}$$

Therefore, the calculation of $\mathbf{f}$ is equivalent to the computation of $\delta\mathbf{u}$. Furthermore, $\delta\mathbf{u}$ can be calculated from the velocity correction at the boundary point, $\delta\mathbf{u}_\mathrm{B}$. As shown in [28], the final expression for $\delta\mathbf{u}_\mathrm{B}$ is given by:

$$\mathbf{AX}=\mathbf{B} \tag{7}$$

where

$$\mathbf{X}=\left\{\delta\mathbf{u}_\mathrm{B}^1,\delta\mathbf{u}_\mathrm{B}^2,\cdots,\delta\mathbf{u}_\mathrm{B}^m\right\}^T;$$

$$\mathbf{A}=\begin{pmatrix}\delta_{11} & \delta_{12} & \cdots & \delta_{1n}\\ \delta_{21} & \delta_{22} & \cdots & \delta_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ \delta_{m1} & \delta_{m2} & \cdots & \delta_{mn}\end{pmatrix}\begin{pmatrix}\delta_{11}^\mathrm{B} & \delta_{12}^\mathrm{B} & \cdots & \delta_{1m}^\mathrm{B}\\ \delta_{21}^\mathrm{B} & \delta_{22}^\mathrm{B} & \cdots & \delta_{2m}^\mathrm{B}\\ \vdots & \vdots & \ddots & \vdots\\ \delta_{n1}^\mathrm{B} & \delta_{n2}^\mathrm{B} & \cdots & \delta_{nm}^\mathrm{B}\end{pmatrix};$$

$$\mathbf{B}=\begin{pmatrix}\mathbf{U}_\mathrm{B}^1\\ \mathbf{U}_\mathrm{B}^2\\ \vdots\\ \mathbf{U}_\mathrm{B}^m\end{pmatrix}-\begin{pmatrix}\delta_{11} & \delta_{12} & \cdots & \delta_{1n}\\ \delta_{21} & \delta_{22} & \cdots & \delta_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ \delta_{m1} & \delta_{m2} & \cdots & \delta_{mn}\end{pmatrix}\begin{pmatrix}\mathbf{u}_1^*\\ \mathbf{u}_2^*\\ \vdots\\ \mathbf{u}_n^*\end{pmatrix}$$

Here, m is the number of boundary points and n is the number of surrounding Cartesian points. $\delta\mathbf{u}_{\mathrm{B}}^{l}\quad (l=1,2,\cdots,m)$ is the unknown velocity correction vector at the boundary point. $\delta_{i'j'}=\mathrm{D}_{ijk}\left(\mathbf{x}_{ijk}-\mathbf{X}_{\mathrm{B}}^{l}\right)\Delta x\Delta y\Delta z$ and $\delta_{i'j'}^{\mathrm{B}}=\mathrm{D}_{ijk}\left(\mathbf{x}_{ijk}-\mathbf{X}_{\mathrm{B}}^{l}\right)\Delta s_l$. Here, $\mathrm{D}_{ijk}\left(\mathbf{x}_{ijk}-\mathbf{X}_{\mathrm{B}}^{l}\right)$ is the delta function that can be expressed as:

$$\mathrm{D}_{ijk}\left(\mathbf{x}_{ijk}-\mathbf{X}_{\mathrm{B}}^{l}\right)=\delta\left(x_{ijk}-X_{\mathrm{B}}^{l}\right)\delta\left(y_{ijk}-Y_{\mathrm{B}}^{l}\right)\delta\left(z_{ijk}-Z_{\mathrm{B}}^{l}\right) \tag{8}$$

$$\delta(r)=\begin{cases}\dfrac{1}{4h}\left(1+\cos\left(\dfrac{\pi|r|}{2h}\right)\right), & |r|\le 2h\\ 0, & |r|>2h\end{cases} \tag{9}$$

where h is the mesh spacing to perform delta function interpolation. Δx, Δy, and Δz are the mesh spacing in the x-, y-, and z-direction, respectively. Δs_l is the area of the boundary element. In current simulation, the triangular element is used to discretize the 3D boundary surface. $\mathbf{U}_{\mathrm{B}}^{l}$ is the boundary velocity. After the boundary velocity correction $\delta\mathbf{u}_{\mathrm{B}}^{l}$ is obtained from Equation (7), the fluid velocity correction $\delta\mathbf{u}$ can be calculate by:

$$\delta\mathbf{u}\left(\mathbf{x}_{ijk}\right)=\sum_{l}\delta\mathbf{u}_{\mathrm{B}}^{l}\left(\mathbf{X}_{\mathrm{B}}^{l}\right)\mathrm{D}_{ijk}\left(\mathbf{x}_{ijk}-\mathbf{X}_{\mathrm{B}}^{l}\right)\Delta s_l \tag{10}$$

Because the relationship between the force density and the fluid velocity correction is $\delta\mathbf{u}=\mathbf{f}\delta t/2\rho$, we can simply compute the force density through:

$$\mathbf{f}\left(\mathbf{x}_{ijk}\right)=2\rho\delta\mathbf{u}\left(\mathbf{x}_{ijk}\right)/\delta t \tag{11}$$

Similarly, the force on the boundary point can be computed from:

$$\mathbf{F}\left(\mathbf{X}_{\mathrm{B}}^{l}\right)=2\rho\delta\mathbf{u}_{\mathrm{B}}^{l}/\delta t \tag{12}$$

This is the force exerted on the fluid. So, its balance is the hydrodynamic force exerted on the boundary of immersed object. In LBM simulation, the macroscopic density, momentum, and pressure can be calculated by:

$$\rho=\sum_{\alpha}\mathrm{f}_{\alpha},\quad \rho\mathbf{u}^{*}=\sum_{\alpha}\mathrm{f}_{\alpha}\mathbf{e}_{\alpha},\quad p=c_s^2\rho \tag{13}$$

To utilize the non-uniform Cartesian mesh in the application of 3D IB–LBM, the efficient LBM solver developed in [28] should be employed. A brief review of this scheme is provided in the following.

2.2. *Efficient three-dimensional lattice Boltzmann method solver on non-uniform Cartesian mesh*

The governing Equation (1) of 3D IB–LBM can be rewritten as:

$$\mathrm{f}_{\alpha}(\mathbf{x},t)=\mathrm{g}_{\alpha}(\mathbf{x}-\mathbf{e}_{\alpha}\delta t,t) \tag{14}$$

$$\mathrm{g}_{\alpha}(\mathbf{x}-\mathbf{e}_{\alpha}\delta t,t)=\mathrm{f}_{\alpha}(\mathbf{x}-\mathbf{e}_{\alpha}\delta t,t-\delta t)+\frac{1}{\tau}\left(\mathrm{f}_{\alpha}^{\mathrm{eq}}(\mathbf{x}-\mathbf{e}_{\alpha}\delta t,t-\delta t)-\mathrm{f}_{\alpha}(\mathbf{x}-\mathbf{e}_{\alpha}\delta t,t-\delta t)\right)+F_{\alpha}\delta t \tag{15}$$

Here, g_{α} is the post-collision state of distribution function. For the non-uniform mesh, $\mathbf{x}-\mathbf{e}_{\alpha}\delta t$ may not be the mesh point. Because the Cartesian mesh is used in IB–LBM, it is easy to evaluate g_{α} at $\mathbf{x}-\mathbf{e}_{\alpha}\delta t$ by using the second-order one-dimensional interpolation. The details of this scheme have been described in [28]. Here, only the final expression for the 3D case is provided.

Setting $g_\alpha\left(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t, z_k - e_{\alpha z}\delta t\right)$ as the post-collision state of distribution function in 3D case, it can be calculated by:

$$g_\alpha\left(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t, z_k - e_{\alpha z}\delta t\right) = c_{1\alpha} g_{t1\alpha} + c_{2\alpha} g_{t2\alpha} + c_{3\alpha} g_{t3\alpha} = \mathbf{CG_t} \tag{16}$$

where $\mathbf{G_t} = \{\mathbf{BG}_1\mathbf{A}, \mathbf{BG}_2\mathbf{A}, \mathbf{BG}_3\mathbf{A}\}^T$, the matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are:

$$\mathbf{A} = \{a_{1\alpha}, a_{2\alpha}, a_{3\alpha}\}^T, \quad \mathbf{B} = \{b_{1\alpha}, b_{2\alpha}, b_{3\alpha}\}, \quad \mathbf{C} = \{c_{1\alpha}, c_{2\alpha}, c_{3\alpha}\} \tag{17}$$

The coefficients in Equation (17) are computed as:

$$a_{1\alpha} = \frac{e_{\alpha x}\delta t\,(e_{\alpha x}\delta t + \Delta x_2)}{\Delta x_1\,(\Delta x_1 - \Delta x_2)}, a_{2\alpha} = \frac{(e_{\alpha x}\delta t + \Delta x_1)\,(e_{\alpha x}\delta t + \Delta x_2)}{\Delta x_1 \Delta x_2}, a_{3\alpha} = \frac{e_{\alpha x}\delta t\,(e_{\alpha x}\delta t + \Delta x_1)}{\Delta x_2\,(\Delta x_2 - \Delta x_1)} \tag{18}$$

$$b_{1\alpha} = \frac{e_{\alpha y}\delta t\,\left(e_{\alpha y}\delta t + \Delta y_2\right)}{\Delta y_1\,(\Delta y_1 - \Delta y_2)}, b_{2\alpha} = \frac{\left(e_{\alpha y}\delta t + \Delta y_1\right)\left(e_{\alpha y}\delta t + \Delta y_2\right)}{\Delta y_1 \Delta y_2}, b_{3\alpha} = \frac{e_{\alpha y}\delta t\,\left(e_{\alpha y}\delta t + \Delta y_1\right)}{\Delta y_2\,(\Delta y_2 - \Delta y_1)} \tag{19}$$

$$c_{1\alpha} = \frac{e_{\alpha z}\delta t\,(e_{\alpha z}\delta t + \Delta z_2)}{\Delta z_1\,(\Delta z_1 - \Delta z_2)}, c_{2\alpha} = \frac{(e_{\alpha z}\delta t + \Delta z_1)\,(e_{\alpha z}\delta t + \Delta z_2)}{\Delta z_1 \Delta z_2}, c_{3\alpha} = \frac{e_{\alpha z}\delta t\,(e_{\alpha z}\delta t + \Delta z_1)}{\Delta z_2\,(\Delta z_2 - \Delta z_1)} \tag{20}$$

with $\Delta x_1 = x_{i-1} - x_i$, $\Delta x_2 = x_{i+1} - x_i$; $\Delta y_1 = y_{j-1} - y_j$, $\Delta y_2 = y_{j+1} - y_j$; and $\Delta z_1 = z_{k-1} - z_k$, $\Delta z_2 = z_{k+1} - z_k$. The matrices $\mathbf{G}_1$, $\mathbf{G}_2$, and $\mathbf{G}_3$ are:

$$\mathbf{G}_1 = \begin{pmatrix} g_\alpha\left(x_{i-1}, y_{j-1}, z_{k-1}\right) & g_\alpha\left(x_i, y_{j-1}, z_{k-1}\right) & g_\alpha\left(x_{i+1}, y_{j-1}, z_{k-1}\right) \\ g_\alpha\left(x_{i-1}, y_j, z_{k-1}\right) & g_\alpha\left(x_i, y_j, z_{k-1}\right) & g_\alpha\left(x_{i+1}, y_j, z_{k-1}\right) \\ g_\alpha\left(x_{i-1}, y_{j+1}, z_{k-1}\right) & g_\alpha\left(x_i, y_{j+1}, z_{k-1}\right) & g_\alpha\left(x_{i+1}, y_{j+1}, z_{k-1}\right) \end{pmatrix}$$
$$\mathbf{G}_2 = \begin{pmatrix} g_\alpha\left(x_{i-1}, y_{j-1}, z_k\right) & g_\alpha\left(x_i, y_{j-1}, z_k\right) & g_\alpha\left(x_{i+1}, y_{j-1}, z_k\right) \\ g_\alpha\left(x_{i-1}, y_j, z_k\right) & g_\alpha\left(x_i, y_j, z_k\right) & g_\alpha\left(x_{i+1}, y_j, z_k\right) \\ g_\alpha\left(x_{i-1}, y_{j+1}, z_k\right) & g_\alpha\left(x_i, y_{j+1}, z_k\right) & g_\alpha\left(x_{i+1}, y_{j+1}, z_k\right) \end{pmatrix} \tag{21}$$
$$\mathbf{G}_3 = \begin{pmatrix} g_\alpha\left(x_{i-1}, y_{j-1}, z_{k+1}\right) & g_\alpha\left(x_i, y_{j-1}, z_{k+1}\right) & g_\alpha\left(x_{i+1}, y_{j-1}, z_{k+1}\right) \\ g_\alpha\left(x_{i-1}, y_j, z_{k+1}\right) & g_\alpha\left(x_i, y_j, z_{k+1}\right) & g_\alpha\left(x_{i+1}, y_j, z_{k+1}\right) \\ g_\alpha\left(x_{i-1}, y_{j+1}, z_{k+1}\right) & g_\alpha\left(x_i, y_{j+1}, z_{k+1}\right) & g_\alpha\left(x_{i+1}, y_{j+1}, z_{k+1}\right) \end{pmatrix}$$

After $g_\alpha\left(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t, z_k - e_{\alpha z}\delta t\right)$ is calculated, it is very easy to get $f_\alpha\left(x_i, y_j, z_k\right)$ by using Equation (14). When the D3Q15 lattice model is adopted, $e_{\alpha x}$, $e_{\alpha y}$, and $e_{\alpha z}$ take either 1 or -1. So, in each direction, it is needed to store six coefficients only. Overall, only 18 coefficients for each mesh point are stored. As shown in [28], the present LBM solver is more efficient than other versions of LBM such as the Taylor series expansion and least squares-based LBM because it needs less computational effort and virtual storage.

2.3. Computational sequence

In this study, we focus on the 3D flows over moving objects with prescribed motion. Therefore, the basic solution procedure of the present solver for the simulations can be outlined as follows:

1. Set initial flow fields and initial location of moving objects.
2. Compute the elements of matrix $\mathbf{A}$ in Equation (7), and obtain $\mathbf{A}^{-1}$ according to the instantaneous position of boundary.

3. Use Equations (14) and (15) to obtain the density distribution function at time $t = t_n$ (initially setting $F_\alpha = 0$). Compute the macroscopic variables using Equation (13).
4. Solve Equation system (7) to obtain the velocity corrections at all boundary points, and use Equation (10) to obtain the fluid velocity corrections.
5. Obtain the force density using Equation (11), and correct the fluid velocity using Equation (6).
6. Compute the equilibrium distribution function using Equation (5).
7. Repeat steps 2–6 for time evolution.

3. SIMULATION OF SOME THREE-DIMENSIONAL FLOWS OVER MOVING OBJECTS

In this section, we will show four numerical examples of applying the improved IB–LBM to simulate 3D flows over moving objects.

3.1. Flow over a rotating sphere

The first example is the laminar flow over a rotating sphere. Induced by the particle–particle or particle–wall collisions, the solid particles may translate and rotate simultaneously in the flow. Generally, a solid particle could be regarded as a sphere in engineering applications associated with particle transport. Hence, the flow over a rotating sphere has interesting phenomenon, and it has received much attention. In general, there are two directions of rotation according to the translation of sphere. One is the transverse direction, where the rotational direction is orthogonal to that of translation. The other is the streamwise direction, where the rotational direction is the same as that of translation. A number of research works have been done for both cases and some attractive features are achieved. Kim and Choi [30] conducted the numerical investigation focusing on the effect of the streamwise rotation on the characteristics of flow over a sphere. In their study, the laminar flow with three different Reynolds numbers is considered. At every Reynolds number, the sphere rotates with several different rotational speeds. The forces exerted on the sphere and vortical structures depend significantly on the Reynolds number and rotational speed.

The streamwise rotation case is also studied in this work. In current simulation, the Reynolds number of $Re = 300$ is selected. Here, the Reynolds number is based on the free stream velocity U_∞ and the sphere diameter D. Two non-dimensional angular velocities of the rotating sphere, $\omega^* = 0.1$ and 0.5, are considered. Here, ω^* is the maximum azimuthal velocity on the sphere surface normalized by the free stream velocity, that is, $\omega^* = \omega D/2U_\infty$. The computational domain is a rectangular box with the size of $25D \times 20D \times 20D$ in the x-, y- and z-direction, respectively. The sphere is located at $(10D, 10D, 10D)$. A non-uniform mesh, which is fine and uniform around the sphere, is taken. The mesh size of $141 \times 121 \times 121$ is used, and the uniform mesh step around the sphere h is 0.025. The surface of the sphere is discretized using triangular elements with 808 vertices. The flow around the stationary sphere at steady state is used as an initial flow field in current simulation.

As pointed out by Kim and Choi [30], the flow over a sphere with the streamwise rotation can be classified into different flow regimes according to the Reynolds number and rotational speed. For the Reynolds number selected in this work, there are two different flow regimes. When $\omega^* = 0.1$, the flow is an unsteady asymmetric, and when $\omega^* = 0.5$, the flow becomes frozen. Their difference can be clearly displayed by using streamlines. Figure 1 shows the temporal evolution of streamlines at $\omega^* = 0.1$. To make comparison,the results of stationary case, which are displayed in Figure 2, are also included. From this figure, it can be found that the flow in the x–y plane maintains symmetric, whereas the flow in the x–z plane becomes periodic. As indicated in [28], such flow phenomenon is recognized as an unsteady planar symmetric. When the sphere starts to make streamwise rotation with $\omega^* = 0.1$, as shown in Figure 1, the symmetry is lost in the x–y plane, while the flow in the x–z plane keeps unsteady. Hence, the flow at this situation is called unsteady asymmetric. Figure 3 shows the temporal evolution of streamlines at $\omega^* = 0.5$. It is noted that the coordinate system rotates with the rotating sphere. It can be found in the figure that the streamlines in a plane just rotates without temporal variation. Hence, the flow at this situation is called frozen.

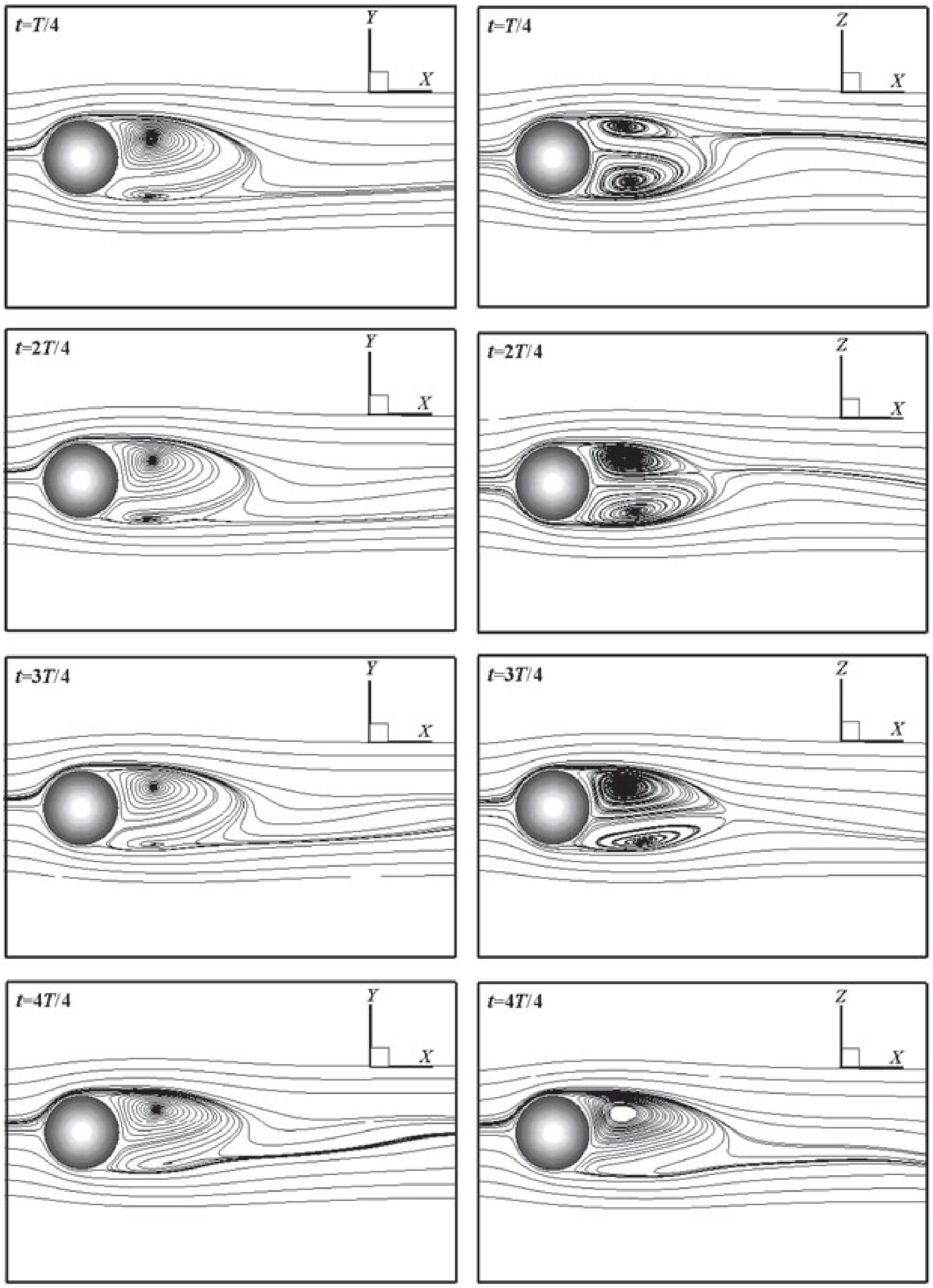


Figure 1. Streamlines for flow over a rotating sphere at $Re = 300$ and $\omega^* = 0.1$ in one complete cycle.

Figure 4 shows the time histories of the drag and lift coefficients at $Re = 300$ with $\omega^* = 0.1$ and 0.5. Here, the drag coefficient C_d is defined as:

$$C_\mathrm{d} = \frac{F_\mathrm{D}}{(1/2)\rho U_\infty^2 S} = \frac{8F_\mathrm{D}}{\rho U_\infty^2 \pi D^2} \tag{22}$$

$$F_\mathrm{D} = -\sum_l F_x\left(\mathbf{X}_B^l\right)\Delta s_l \tag{23}$$

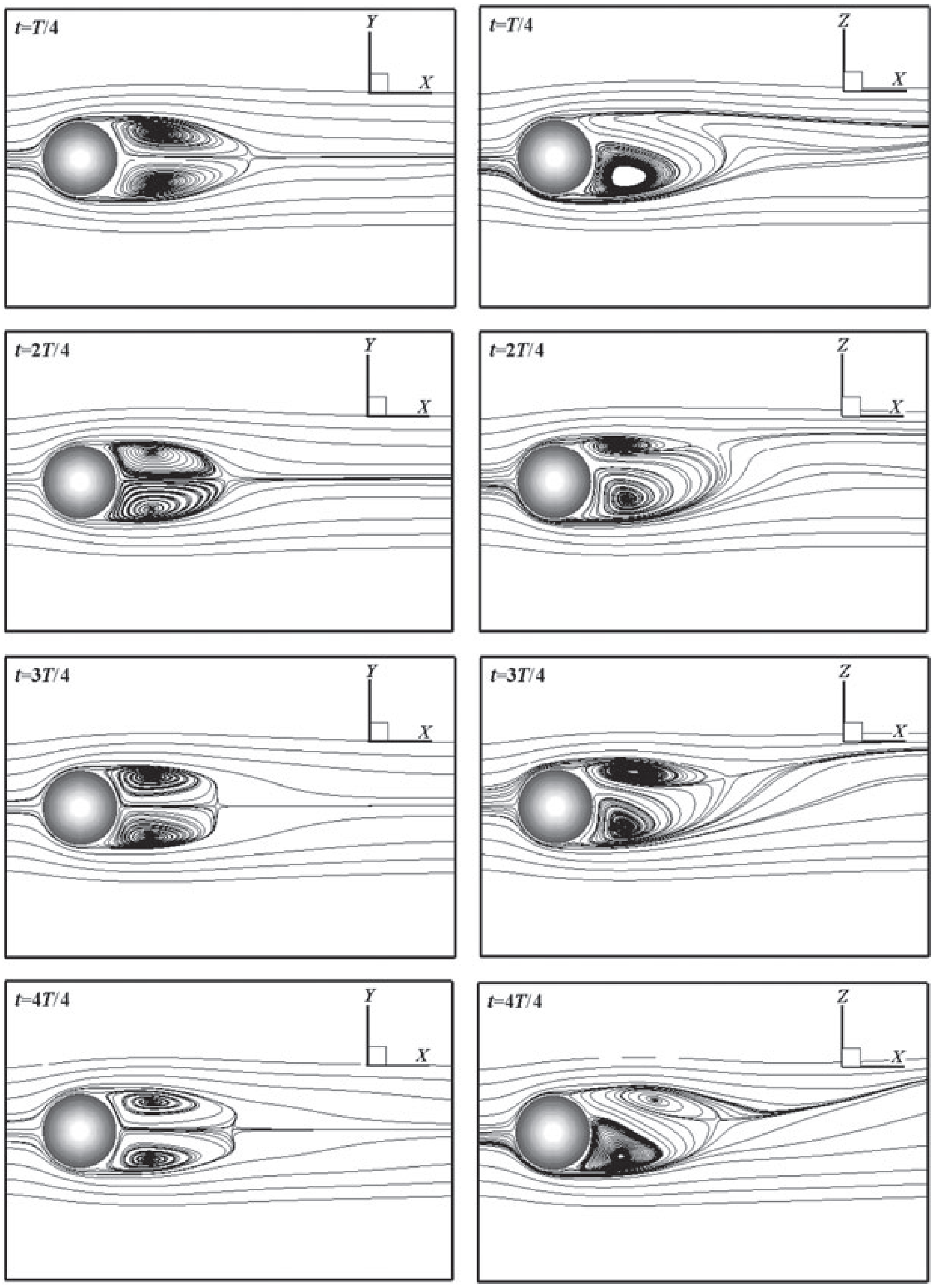


Figure 2. Streamlines for flow over a stationary sphere at $Re = 300$ in one complete cycle.

and the lift coefficient C_l is defined as $C_l = \sqrt{C_y^2 + C_z^2}$, which is the same as used in [30]. Similar to the drag coefficient C_d expressed in Equation (22), C_y and C_z can be calculated by:

$$C_y = \frac{8F_Y}{\rho U_\infty^2 \pi D^2}, \quad C_z = \frac{8F_Z}{\rho U_\infty^2 \pi D^2} \tag{24}$$

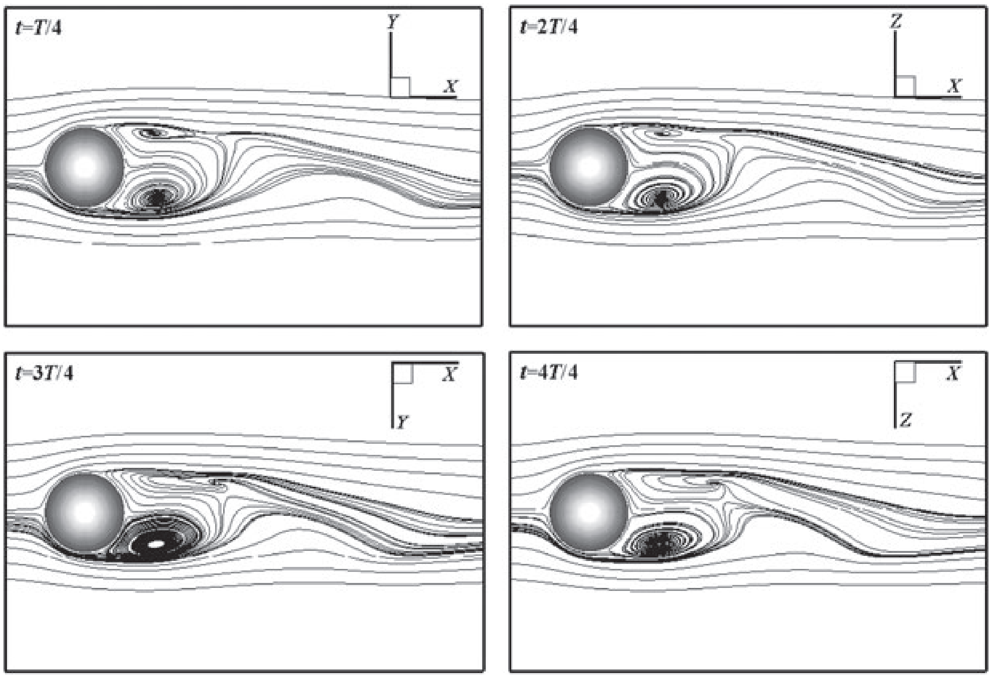


Figure 3. Streamlines for flow over a rotating sphere at $Re = 300$ and $\omega^* = 0.5$ in one complete cycle (coordinate system rotates with sphere).

$$F_Y = -\sum_l F_y\left(\mathbf{X}_{\mathrm{B}}^l\right)\Delta s_l, \quad F_Z = -\sum_l F_z\left(\mathbf{X}_{\mathrm{B}}^l\right)\Delta s_l \tag{25}$$

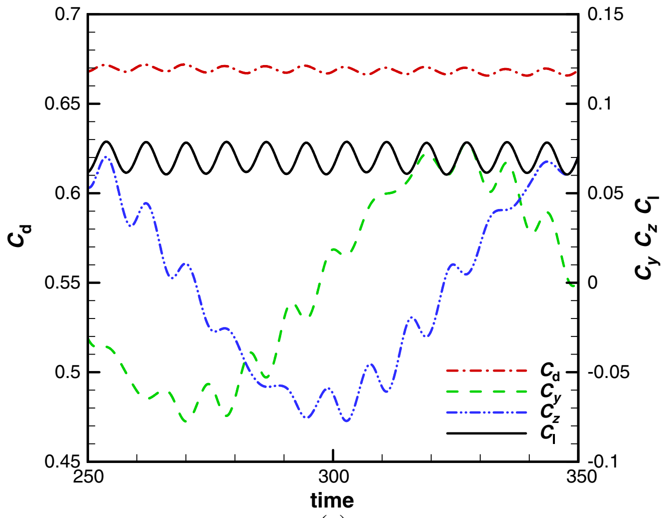
Here $F_x\left(\mathbf{X}_{\mathrm{B}}^l\right)$, $F_y\left(\mathbf{X}_{\mathrm{B}}^l\right)$, and $F_z\left(\mathbf{X}_{\mathrm{B}}^l\right)$ are the boundary forces on the sphere surface in the x-, y-, and z-direction, respectively. They can be simply calculated by using Equation (12). The difference between unsteady asymmetric and frozen flows can be further verified from this figure. For the unsteady asymmetric case (as shown in Figure 4(a)), the drag and lift coefficients vary periodically. In contrast, for the frozen case (as shown in Figure 4(b)), the magnitudes of drag and lift coefficients keep constant. Figure 5 plots the phase diagram of C_y and C_z for two rotating speeds, and the results of Kim and Choi [30] are also included. At $\omega^* = 0.1$, the phase diagram is a curve turning around the origin (Figure 5(a)). At $\omega^* = 0.5$, the phase diagram becomes a circle (Figure 5(b)) because the flow is frozen. It is clear from the figure that the overall behavior of the current phase diagram is in line with that obtained by Kim and Choi [30].

Figure 6 shows the 3D vortical structures due to the rotation of the sphere at $Re = 300$ with $\omega^* = 0.1$ and 0.5. To calculate the 3D vortical structures, the λ_2-method of Jeong and Hussain [31] is employed in this study, unless otherwise specified. As compared with the stationary sphere case, as shown in Figure 7, the vortical structures are significantly modified, and the flows lose the planar symmetry because of the streamwise rotation of the sphere.

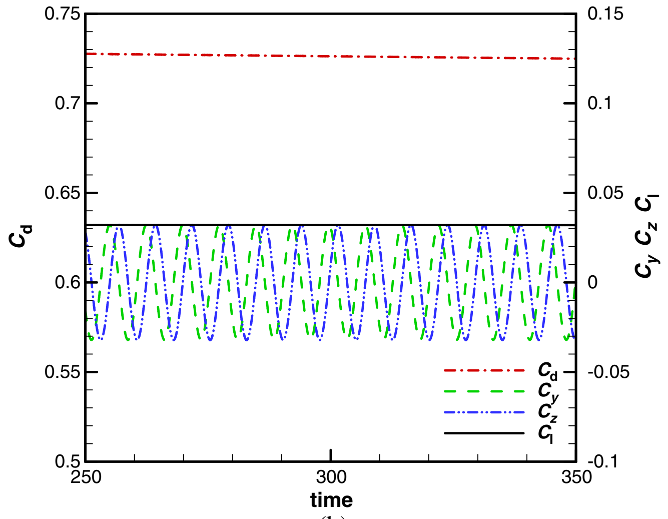
Based on the obtained results, it is clear that the improved IB–LBM can be employed to simulate 3D flows around moving objects with good accuracy. The expected flow patterns can be well captured. To further validate the method and make comparison with the case of rotating sphere as well, the laminar flow over a rotating torus is also simulated by the improved IB–LBM.

3.2. Flow over a rotating torus

As pointed out by Sheard *et al.* [32], the behaviors of vortical structure behind the torus highly depend on its aspect ratio (Ar). As shown in Figure 8, the aspect ratio is defined as $Ar = D/d$,



(a)



(b)

Figure 4. Time histories of the drag and lift coefficients for flows over a rotating sphere at $Re = 300$, (a) $\omega^* = 0.1$ and (b) $\omega^* = 0.5$.

where D is the mean torus diameter and d is the cross-sectional diameter. It is noted that the hole in the center of the torus starts to appear when $Ar = 1$. When Ar is small (less than 4) [32], the wake behind the torus is similar to that of a sphere. In order to make comparison with the results of a rotating sphere case, the flow over a rotating torus with small aspect ratios is simulated.

In this study, two aspect ratios, that is, $Ar = 0.5$ and 2, are considered. The computational domain is a rectangular box and its size is $25d \times 20d \times 20d$ in the x-, y- and z-direction, respectively. The torus is located at $(10d, 10d, 10d)$. A non-uniform mesh, which is fine and uniform around the torus, is taken. Two selected non-dimensional angular velocities, based on the free stream velocity U_∞ and mean torus diameter D, are taken as $\omega^* = 0.1$ and 0.5, which are the same as in the case of a rotating sphere. The flows around stationary torus at steady state are used as initial flow fields for present unsteady simulation.

3.2.1. Torus without hole ($Ar = 0.5$). For $Ar = 0.5$, the Reynolds number $Re = 180$, based on the free stream velocity U_∞ and cross-sectional diameter d, is selected in current simulation. As

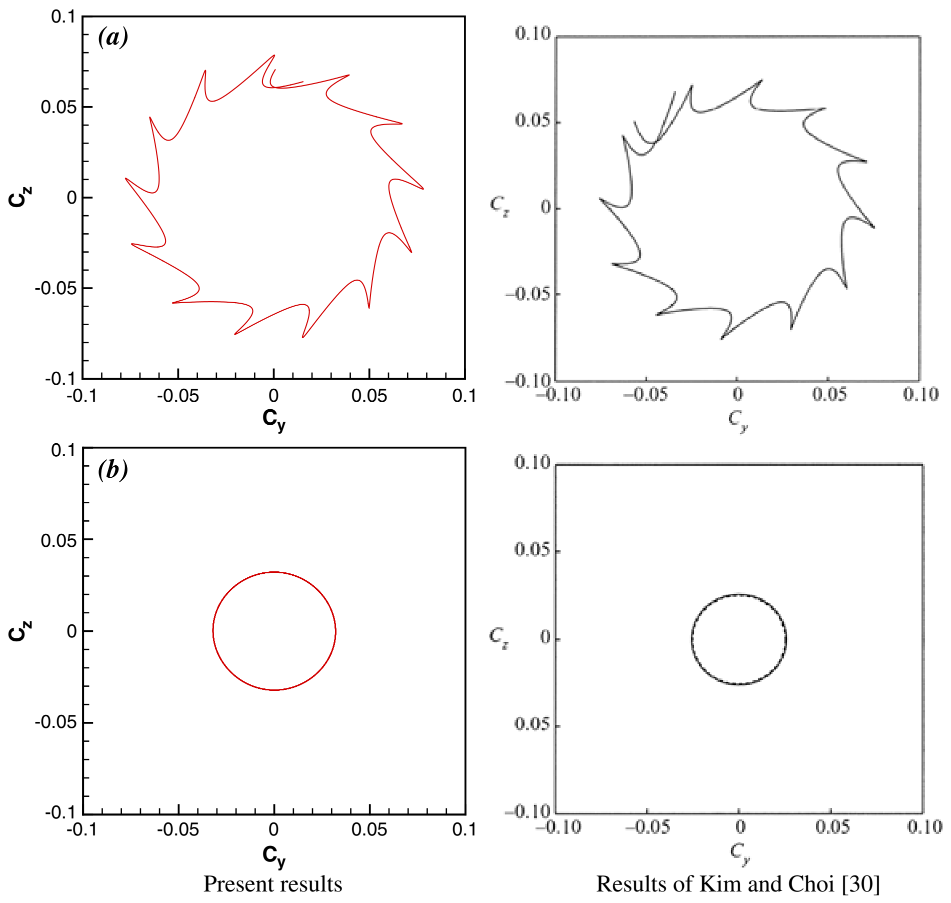


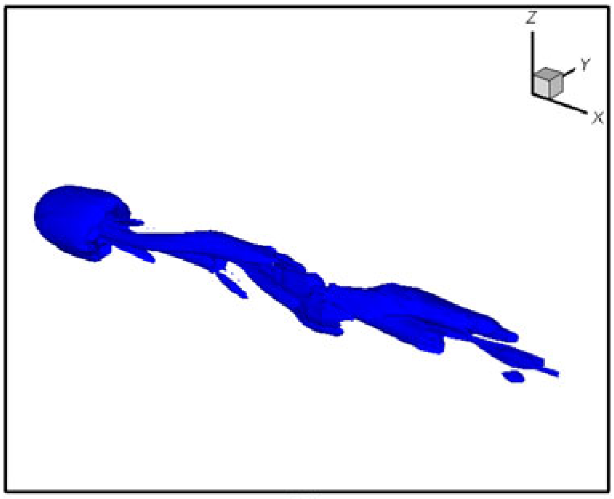
Figure 5. Phase diagram of C_y and C_z for flows over a rotating sphere at $Re = 300$, (a) $\omega^* = 0.1$ and (b) $\omega^* = 0.5$.

indicated in [28], for the stationary case, the flow is an unsteady planar symmetric at this Reynolds number, and there is no hole in the torus at this aspect ratio. The resultant flow behavior has a similarity to that over a sphere. Therefore, for the rotating case, we expect that the wake behind the torus can also demonstrate a similarity to that over a rotating sphere that was shown in the previous section. In the current simulation, the mesh size is chosen as 101×101×101. The uniform mesh spacing around the torus is taken as 0.05. The surface of the torus is discretized using triangular elements with 458 vertices.

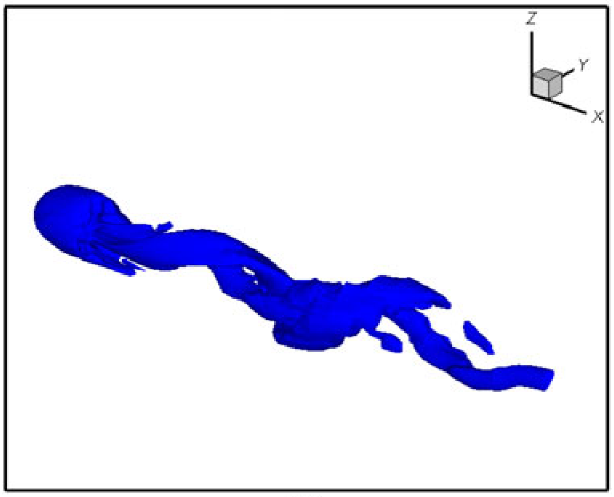
Figure 9 shows the temporal evolution of streamlines at $\omega^* = 0.1$. Because the flow is an unsteady planar symmetric for the stationary case, as shown in Figure 10, it can be found that although the flow in the $x - y$ plane is asymmetric, the symmetry of flow is still maintained in the $x - z$ plane. However, as displayed in Figure 9, the flow in the x–z plane also becomes asymmetric if the torus rotates with $\omega^* = 0.1$. Figure 11 provides the temporal evolution of streamlines at $\omega^* = 0.5$. It should be indicated that the coordinate system rotates with the rotating torus. From the figure, it is clear that the streamlines in a plane just rotates without temporal variation. As compared with the results of the rotating sphere, as shown in Figures 1 and 3, two different flow regimes, unsteady asymmetric flow (Figure 9) and frozen flow (Figure 11), also appear for the case of the rotating torus with $Ar = 0.5$.

Figure 12 plots the resultant time histories of the drag and lift coefficients. Note that the definition of the drag coefficient for a torus is different from that of a sphere. According to Sheard *et al.* [33], C_d is defined as:

$$C_\mathrm{d} = \frac{2F_\mathrm{D}}{\rho A_\mathrm{frontal} U_\infty^2} \tag{26}$$



(a)



(b)

Figure 6. Three-dimensional vortical structures of rotating sphere at $Re = 300$, (a) $\omega^* = 0.1$ and (b) $\omega^* = 0.5$.

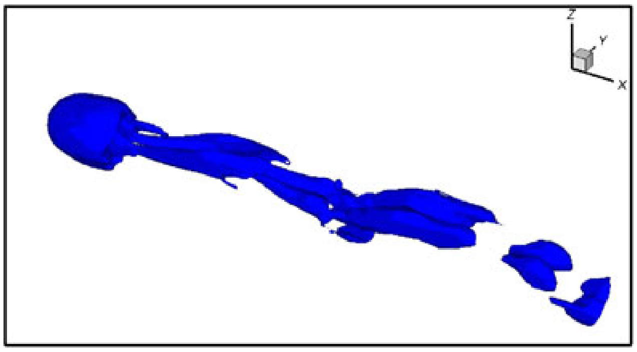


Figure 7. Three-dimensional vortical structures of stationary sphere at $Re = 300$.

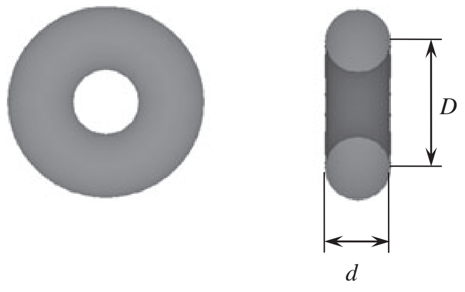


Figure 8. Dimensions of the torus.

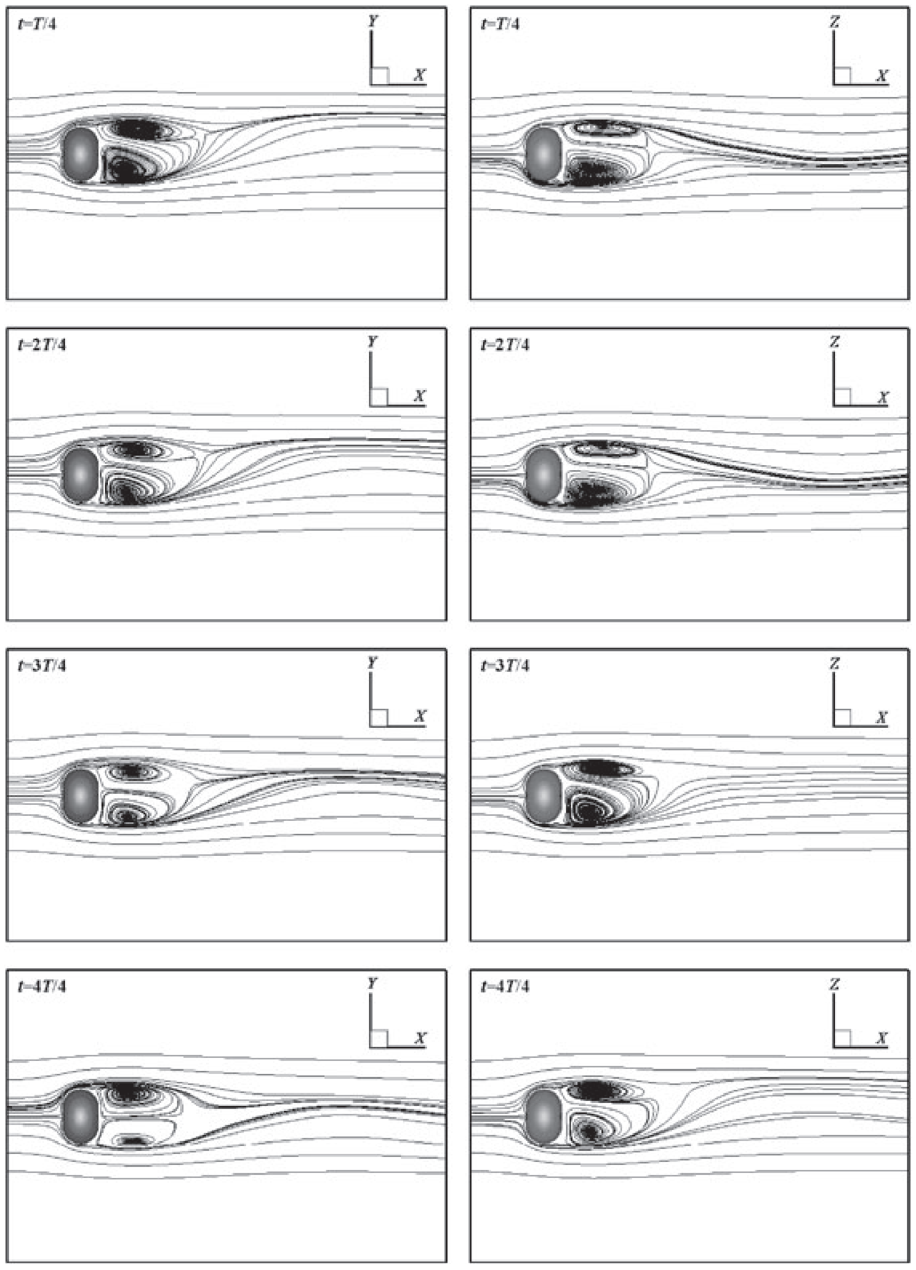


Figure 9. Streamlines for flow over a rotating torus with $Ar = 0.5$ at $Re = 180$ and $\omega^* = 0.1$ in one complete cycle.

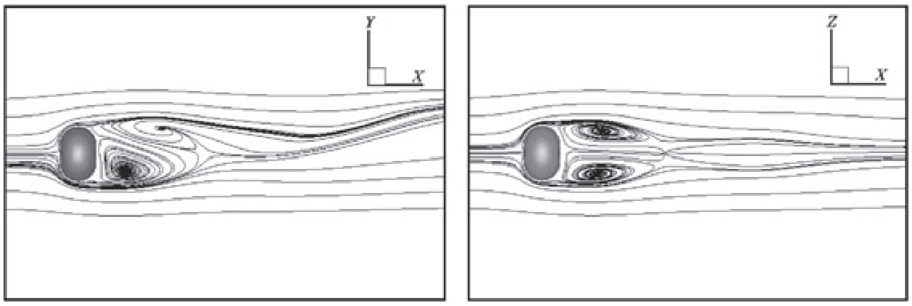


Figure 10. Streamlines for flow over a stationary torus with $Ar = 0.5$ at $Re = 180$.

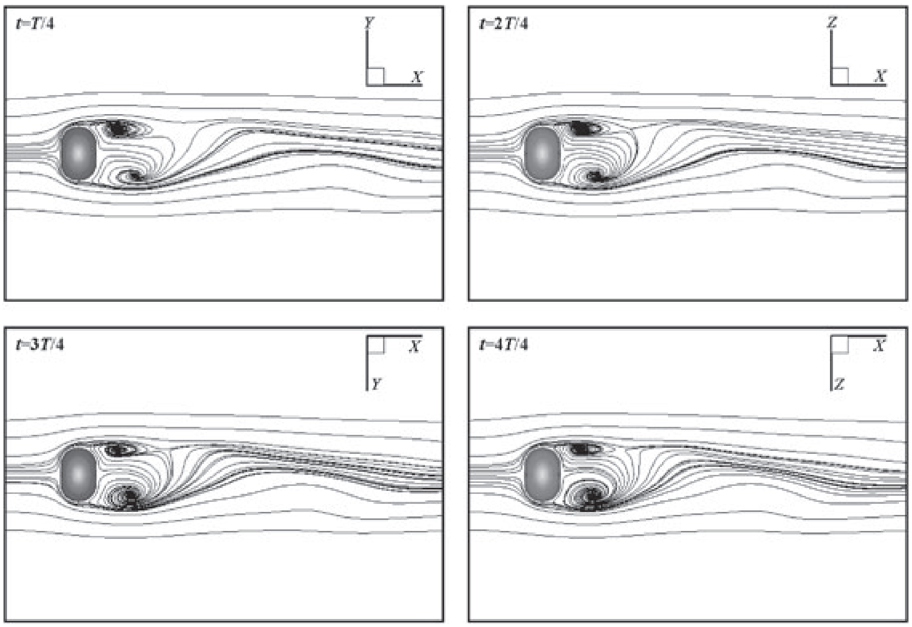


Figure 11. Streamlines for flow over a rotating torus with $Ar = 0.5$ at $Re = 180$ and $\omega^* = 0.5$ in one complete cycle (coordinate system rotates with torus).

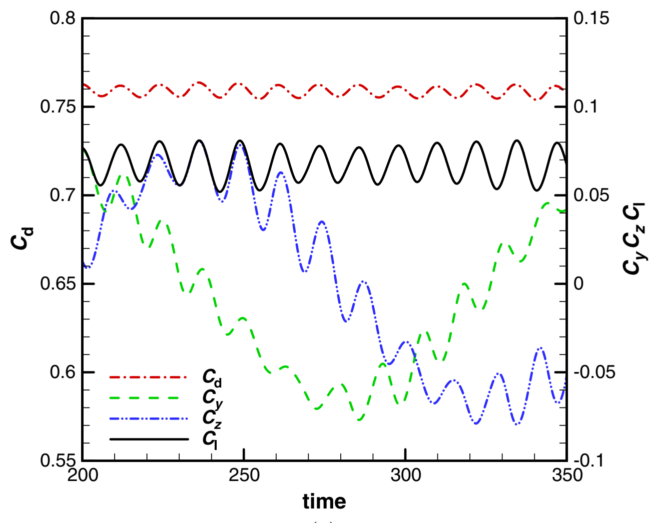
where A_{frontal} is the projected frontal area of torus, which is a function of Ar:

$$A_{\text{frontal}} = \begin{cases} \dfrac{4}{\pi\,(Ar^2 + 2Ar + 1)} & 0 \leq Ar \leq 1 \\ \dfrac{1}{\pi Ar} & Ar > 1 \end{cases} \tag{27}$$

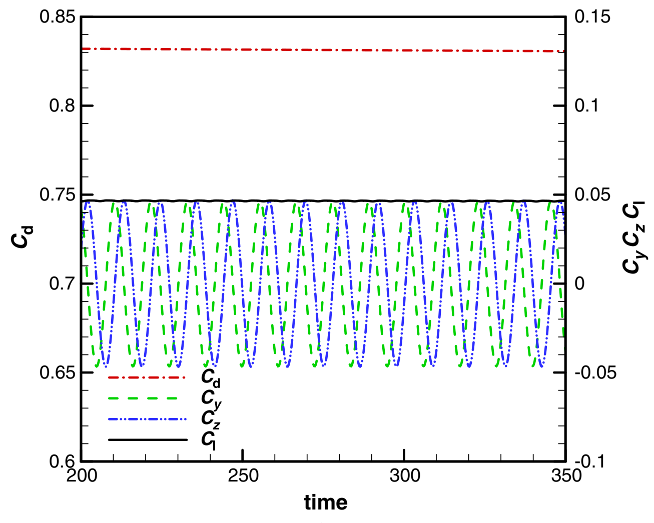
Same as with the case of the rotating sphere, the lift coefficient C_l is defined as $C_l = \sqrt{C_y^2 + C_z^2}$. Similarly, C_y and C_z can be calculated by:

$$C_y = \frac{2F_Y}{\rho A_{\text{frontal}} U_\infty^2}, \quad C_z = \frac{2F_Z}{\rho A_{\text{frontal}} U_\infty^2} \tag{28}$$

From Figure 12, it is found that the variation of coefficients has a similarity with that of the rotating sphere as shown in Figure 4. The only difference is that the frequency of lift coefficient for the rotating sphere is higher than that for the rotating torus. One possible reason is that because the area of vertical section (x-direction) in torus is larger, it takes a longer time for fluid to flow along the y- and z-directions on the surface of torus. Figure 13 illustrates the phase diagram of C_y and C_z for $\omega^* = 0.1$ and 0.5. Similar diagram for the rotating sphere case is shown in Figure 5. Based on



(a)



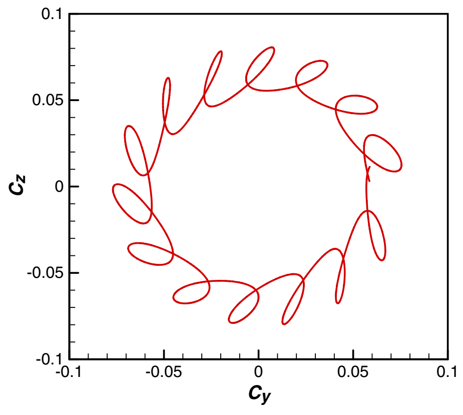
(b)

Figure 12. Time histories of the drag and lift coefficients for flows over a rotating torus with $Ar = 0.5$ at $Re = 180$, (a) $\omega^* = 0.1$ and (b) $\omega^* = 0.5$.

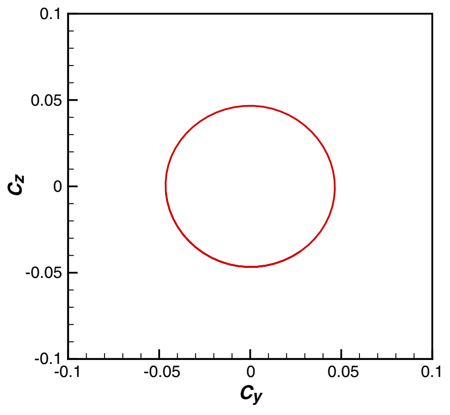
the results in Figures 12 and 13, it well demonstrates the unsteady asymmetric flow (Figures 12(a) and 13(a)) and the frozen flow (Figure 12(a) and 13(b)) patterns. Figure 14 shows the 3D vortical structures around torus with rotation. The flow patterns are similar to those of the rotating sphere case as shown in Figure 6.

3.2.2. Torus with a hole ($Ar = 2$). For $Ar = 2$, the Reynolds number of $Re = 120$ is selected in current simulation. The mesh size is chosen as $101 \times 121 \times 121$. The uniform mesh spacing around the torus is taken as 0.05. The surface of torus is discretized using triangular elements with 776 vertices.

Because there is a hole in the center of the torus at this Ar, the flow behavior exhibits a great difference from that over a sphere or a torus with $Ar < 1$. Figures 15 and 16 display the temporal evolution of streamlines at $\omega^* = 0.1$ and 0.5, respectively. Note that the coordinate system rotates with the rotating torus for $\omega^* = 0.5$. It can be seen from the figures that the flows are an unsteady

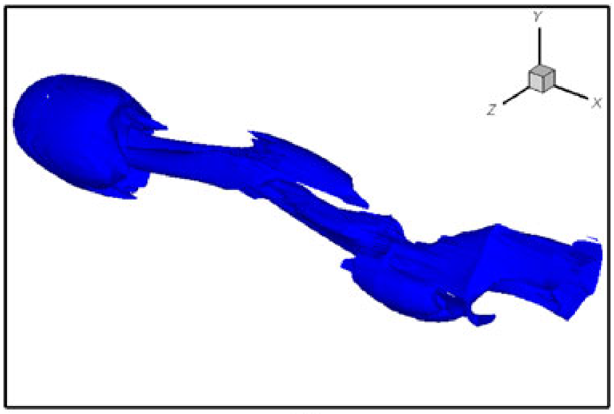


(a)

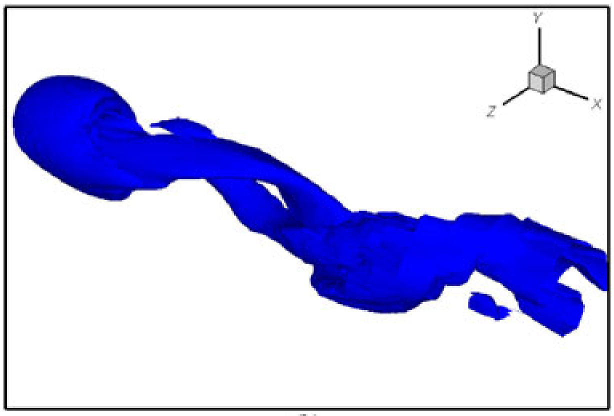


(b)

Figure 13. Phase diagram of C_y and C_z for flows over a rotating torus with $Ar = 0.5$ at $Re = 180$, (a) $\omega^* = 0.1$ and (b) $\omega^* = 0.5$.



(a)



(b)

Figure 14. Three-dimensional vortical structures of rotating torus with $Ar = 0.5$ at $Re = 180$, (a) $\omega^* = 0.1$ and (b) $\omega^* = 0.5$.

asymmetric for two rotating speeds of 0.1 and 0.5. This phenomenon can be further confirmed by the time histories of the drag and lift coefficients shown in Figure 17. Because of the existence of a hole in the center of the torus, the flow can go through the torus. Such flow leak may destroy the formation of frozen flow at $\omega^* = 0.5$. Figure 18 plots the corresponding 3D vortical structures. As

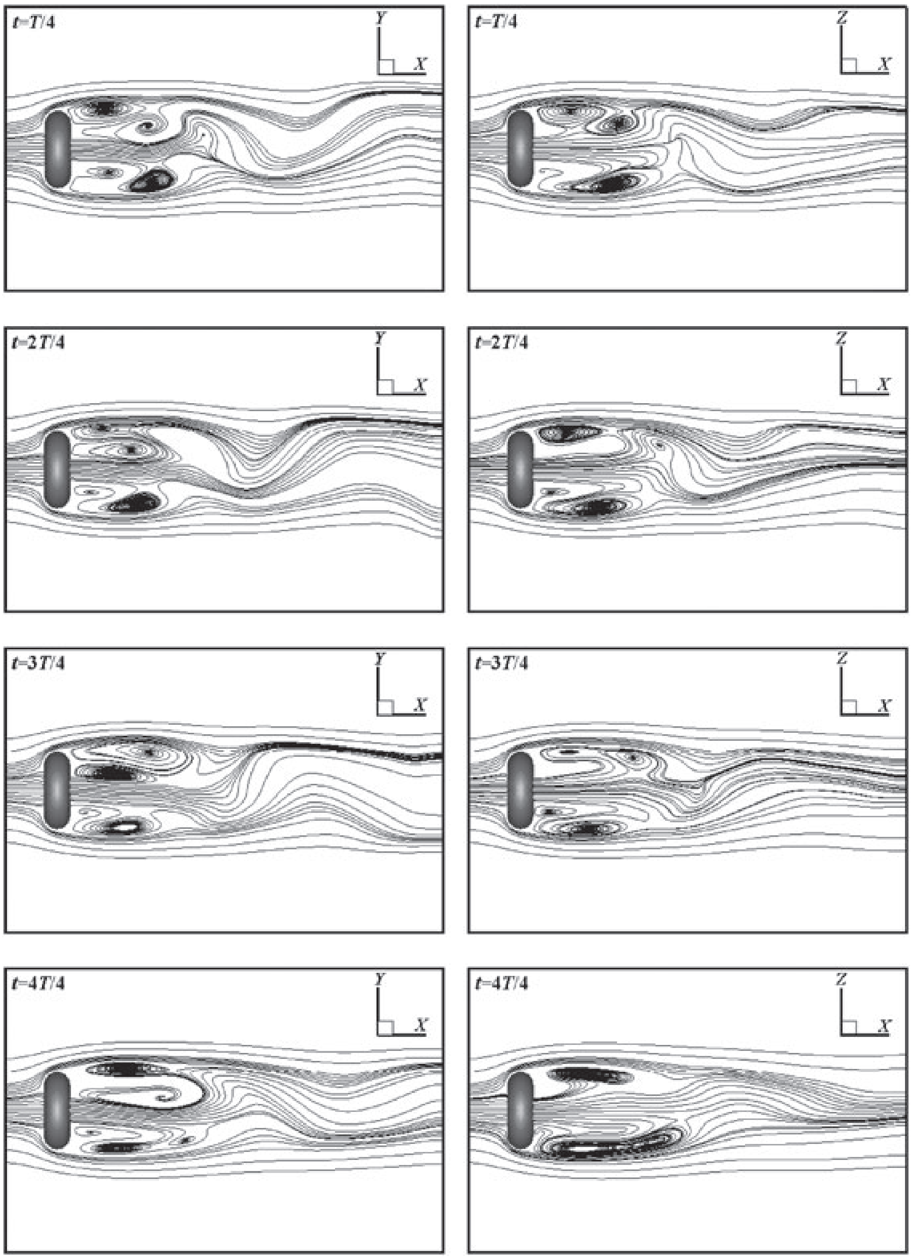


Figure 15. Streamlines for flow over a rotating torus with $Ar = 2$ at $Re = 120$ and $\omega^* = 0.1$ in one complete cycle.

compared with the results in Figures 6 and 14, only an unsteady asymmetric structure can be found in Figure 18.

To demonstrate the capability of the present solver for modeling 3D flows over moving objects with highly complex geometries, the simulations of fish swimming and dragonfly flight are performed in the following sub-sections.

3.3. *Fish swimming*

The locomotion of aquatic animals has drawn a great attention to biologists and engineers. Using the jet stream propulsion effectively, the aquatic animals could achieve remarkable propulsive efficiency

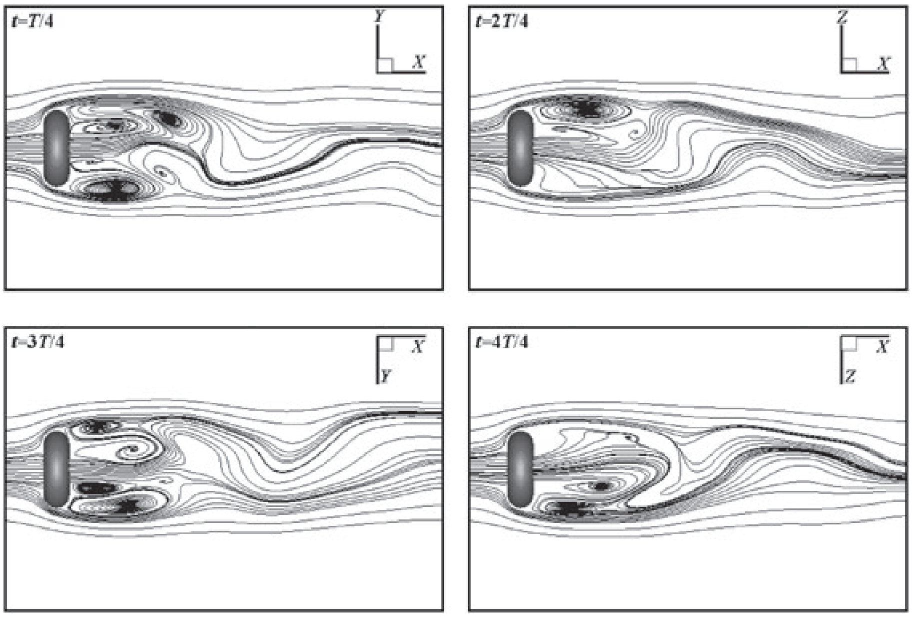


Figure 16. Streamlines for flow over a rotating torus with $Ar = 2$ at $Re = 120$ and $\omega^* = 0.5$ in one complete cycle (coordinate system rotates with torus).

as compared with the man-made propulsors and propellers. The work is attributed to the fluid dynamics of undulatory swimming, in which a transverse wave propagates along the body. Such swimming pattern has been observed to be the most effective movement of swimming propulsion employed by a great number of aquatic animals. Among them, the fish swimming has been extensively investigated over the past century. It includes the theoretical study [34], experimental study [35], and numerical study [36, 37].

In this study, the improved IB–LBM is applied for modeling the 3D fish swimming with prescribed kinematics. Here, the fish body shape representing a *RoboTuna* [37] is employed, as shown in Figure 19. Assuming the length of the fish body (from the head to the peduncle, without the caudal fin) to be L_b, the profile of the *RoboTuna* can be described as:

$$z(x)/L_b = \pm 0.152 \tanh(6x/L_b + 1.8), \quad -0.3 \le x/L_b \le 0.1 \tag{29}$$

$$z(x)/L_b = \pm(0.075 - 0.076 \tanh(7x/L_b - 3.15)), \quad 0.1 < x/L_b \le 0.35 \tag{30}$$

$$z(x)/L_b = \pm(1.749 \tanh(x/L_b) - 3.331 \tanh(2x/L_b) + 1.976 \tanh(3x/L_b)), \quad 0.35 < x/L_b \le 0.7 \tag{31}$$

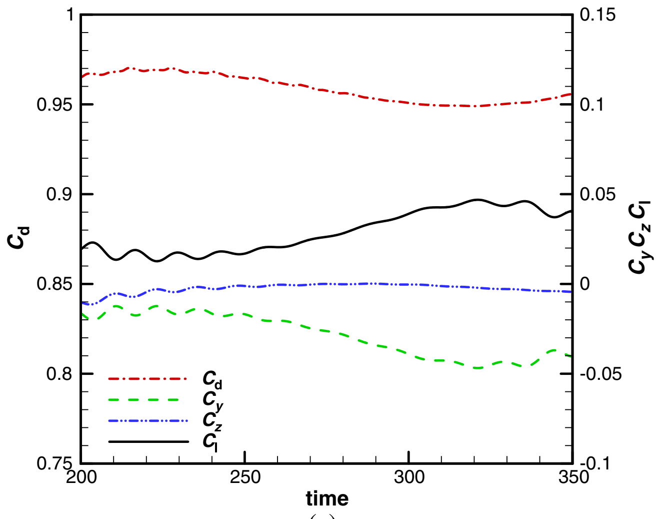
At each horizontal position x, the body sections are assumed to be elliptical with a major-to-minor ratio of $AR = 1.5$, where the major axis corresponds to the height of the body. The caudal fin has chordwise sections of NACA0040 shape. The leading edge and trailing edge profiles are given by:

$$x(z)_{LE}/L_b = 39.543|z/L_b|^3 - 3.685(z/L_b)^2 + 0.636|z/L_b| + 0.7 \tag{32}$$

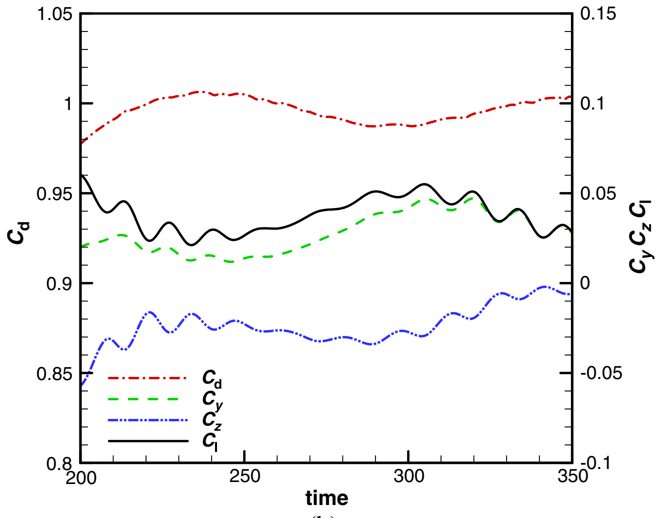
$$x(z)_{TE}/L_b = -40.74|z/L_b|^3 + 9.666(z/L_b)^2 + 0.77 \tag{33}$$

where $-0.15 \le z/L_b \le 0.15$. As a result, the length of the caudal fin L_{cf} can be calculated by $L_{cf} = (x(0.15)_{TE} - x(0)_{LE})L_b$. Hence, the fish length is $L = L_b + L_{cf}$.

In the present simulation, the fish is considered as a straight-line swimming flexible body with a constant speed U_∞ in the negative x-direction. For simplicity, the bending of the fish happens only



(a)



(b)

Figure 17. Time histories of the drag and lift coefficients for flows over a rotating torus with $Ar = 2$ at $Re = 120$, (a) $\omega^* = 0.1$ and (b) $\omega^* = 0.5$.

within the x–y plane. The kinematics for the fish used here is in the form of a traveling backbone wave with the largest wave amplitude at the fish tail. The backbone waveform can be written as:

$$y\,(x,t)\,/L = a\,(x/L)\sin\left(k_{\mathrm{w}}x - \omega t\right) \tag{34}$$

$$a\,(x/L) = a_0 + a_1 x/L + a_2\,(x/L)^2 \tag{35}$$

where $k_{\mathrm{w}} = 2\pi/\lambda$ is the wavenumber of body undulations that corresponds to a wavelength λ, ω is the angular frequency, and $a\,(x/L)$ is the amplitude envelope with the coefficients $a_0 = 0.02$, $a_1 = -0.08$ and $a_2 = 0.16$. With this amplitude envelope, the maximum displacement is at the fish tail with $a_{\max} = 0.1$ that gives $y_{\max} = 0.1L$.

Based on the fish length L and swimming speed U_∞, the Reynolds number for the fish swimming can be defined as $Re = U_\infty L/\upsilon$. Using the maximum lateral excursion of the tail $A = 2y_{\max}$ and

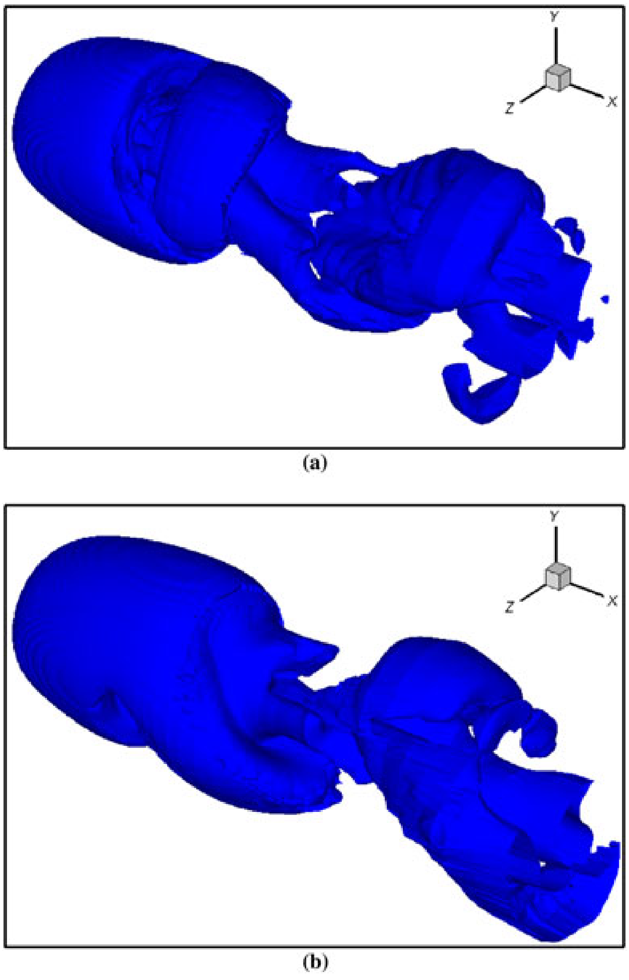


Figure 18. Three-dimensional vortical structures of rotating torus with $Ar = 2$ at $Re = 120$, (a) $\omega^* = 0.1$ and (b) $\omega^* = 0.5$.

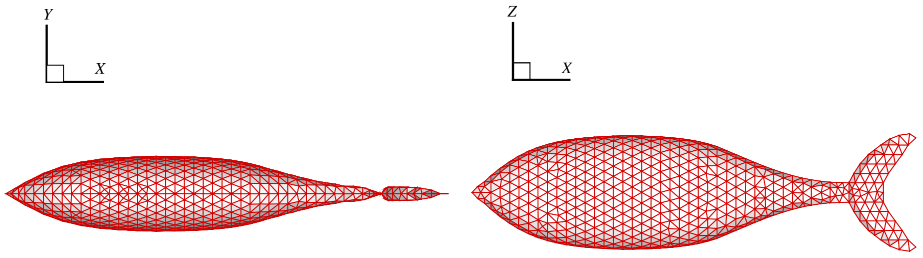


Figure 19. Computational geometric form of *RoboTuna*.

the tail beat frequency f, the Strouhal number can be defined as $St = Af/U_\infty = 2fy_{\max}/U_\infty$. Because $f = \omega/2\pi$, ω can be written as $\omega = \pi St U_\infty/y_{\max}$. Here, $Re = 4000$ is selected to carry out the numerical simulation. Five different Strouhal numbers are chosen, $St = 0.0$, 0.3, 0.5, 0.7, and 0.8. Here, $St = 0.0$ means that the fish is rigid. In addition, $\lambda = 0.95L$ is used. The parameters used in this work are the same as those used by Borazjani and Sotiropoulos [38]. The only difference is that they employed a fish body representing a mackerel. The computational domain in the current simulation is a rectangular box with a range of $[-2L, 6L]\times[-L, L]\times[-L, L]$ in the

x-, y- and z-direction, respectively, which is similar to the work of Borazjani and Sotiropoulos [38]. A non-uniform mesh, which is fine and uniform around the fish body, is taken. The uniform mesh step around the fish body h is 0.05. The surface of fish body is discretized using triangular elements with 925 vertices.

Figure 20 plots the time histories of the force coefficient at $Re = 4000$ with different Strouhal numbers. Here, the coefficient is defined as:

$$C_F = -C_\mathrm{d} = -F_\mathrm{D}/\rho U_\infty^2 L^2 \tag{36}$$

where F_D is the drag force of fish body, which can be calculated using Equation (12). It is noted that the values of C_F in Figure 20 are scaled with the drag coefficient calculated for the rigid body fish. The positive and negative values indicate the thrust-type and drag-type forces, respectively. For the small St (= 0.3, 0.5), the force is in the drag-type regime throughout the whole swimming cycle. As St is increased to 0.7, the excursions of force into the thrust-type regime appear. Further increase of St (= 0.8) leads to longer and larger amplitude excursions into the thrust-type regime, and the mean net force becomes positive. It is noted that the variation trend of force coefficient as the function of St in the current simulation is similar to that of Borazjani and Sotiropoulos [38]. The discrepancy is possibly caused by the use of different shape of fish body.

As indicated by Borazjani and Sotiropoulos [38], the wake behind the fish can be classified as the drag type or thrust type with respect to the direction of the flow between the wake vortices. For the drag-type wake, the regular Karman Vortex Street can be found. In contrast, the reverse Karman Vortex Street appears in the thrust-type wake. Such representative wake patterns are illustrated in Figure 21, which plots the streamlines at the x–y plane of symmetry and the vorticity contours. In the meantime, it can also be found in the figure that the wake has a single-row pattern for the regular Karman Vortex Street, and double-row pattern for the reverse Karman Vortex Street. As pointed out by Borazjani and Sotiropoulos [38], the single-row wake pattern implies that it is confined within a relatively narrow parallel strip centered on the axis of the fish body. For the double-row wake pattern, it is characterized by the lateral divergence and spreading of the vortices away from the body in a wedge-like arrangement. These two distinctly different wake patterns can be further verified through the illustration of 3D vortical structures, as shown in Figure 22. It is noted that the same wake patterns are also found by Borazjani and Sotiropoulos [38].

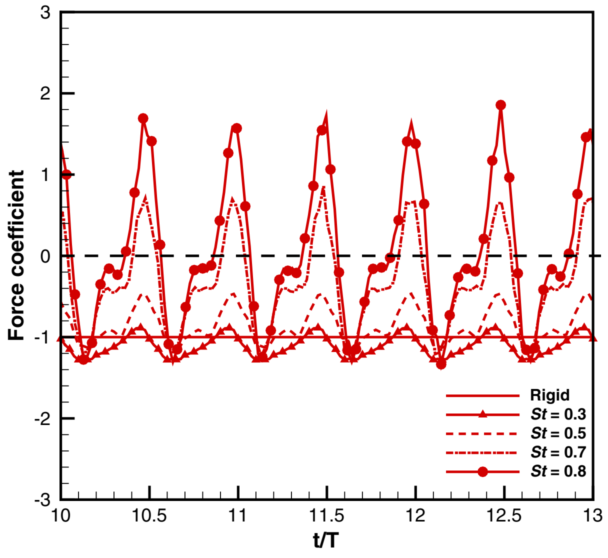


Figure 20. Time histories of the force coefficient for fish swimming at $Re = 4000$.

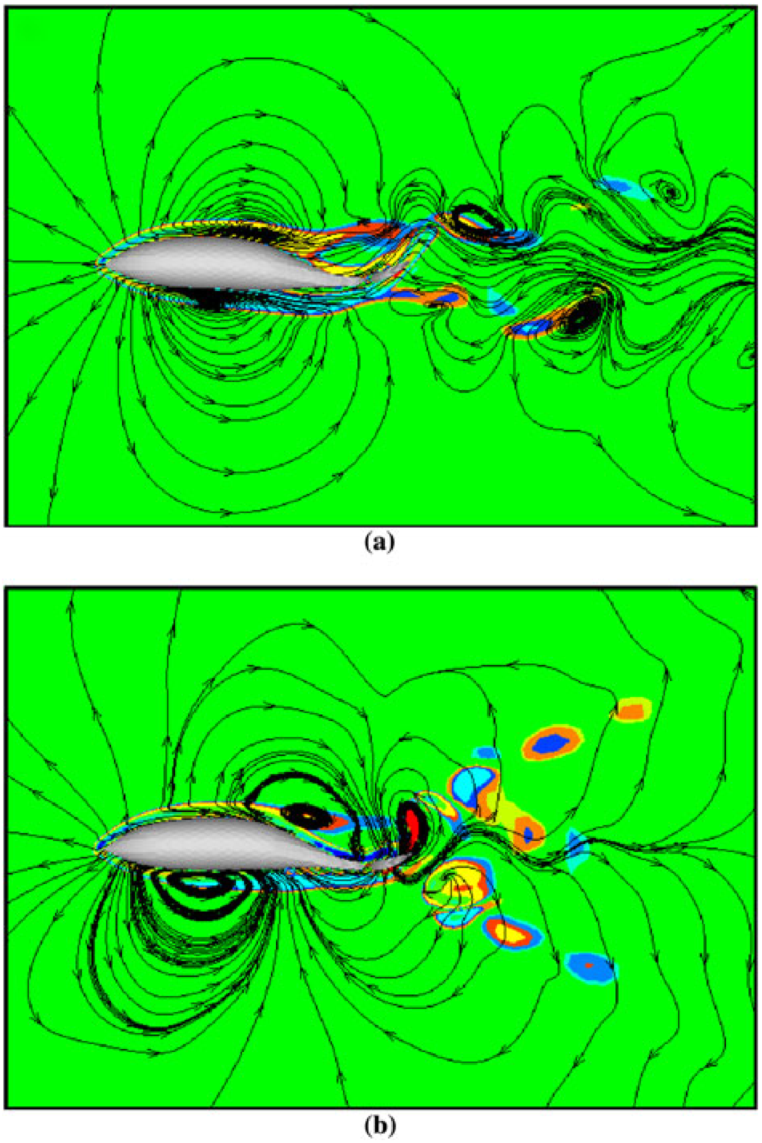


Figure 21. Streamlines (solid lines) with vorticity contours (colored flood) for fish swimming at $Re = 4000$, (a) $St = 0.3$ and (b) $St = 0.8$.

3.4. *Dragonfly flight*

From the unsteady aerodynamic mechanisms, the flapping flying insects are well known for their impressive flight agility, maneuverability, and endurance, including low-speed hovering, effective gliding flight, and sudden variation in flight speed and altitude. Because of high complexity of wing–wing interaction and wing–body interaction [39–41], it is difficult for conventional numerical methods to simulate insect flight. To examine the capability of the present solver for handling such complicated motion, the simulation of dragonfly flight is conducted.

The shape of a dragonfly wing is based on the image of *Sympetrum corruptum* from a website, and the dragonfly body is simply modeled using an ellipse with different major-to-minor ratios (*AR*) at different body sections, as shown in Figure 23. The actual kinematics of a dragonfly is very complicated. For simplicity, a relatively simple representation of the wing kinematics is taken. It is assumed that each pair of wings undergoes a sinusoidal pitching–rolling motion. The pitching is along a spanwise axis that is located at 10% of wing chord. The rolling is along a streamwise axis that is situated at the inner (closer to body) tip of the wing. Here, the wing motion and arrangement

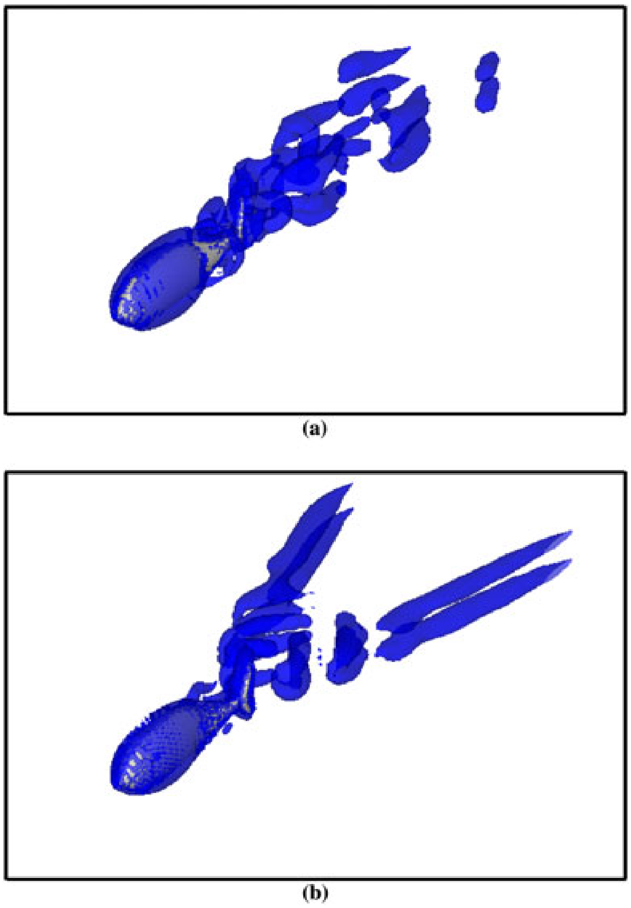


Figure 22. Three-dimensional vortical structures for fish swimming at $Re = 4000$, (a) $St = 0.3$ and (b) $St = 0.8$.

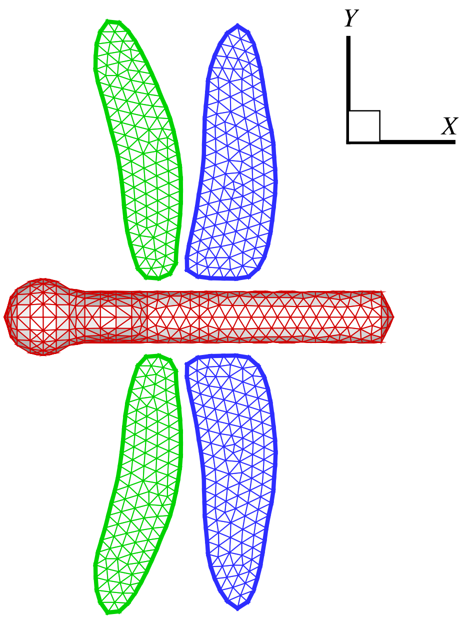


Figure 23. Computational model of dragonfly.

are similar to the work of Mittal *et al.* [42]. In addition, the dragonfly is set to move forward in a straight line along the negative x-direction with constant speed U_∞. As shown in Figure 24, the pitching–rolling motion can be expressed as:

$$\theta = \theta_0 \sin(2\pi t/T + \phi) \tag{37}$$

$$\alpha = \alpha_0 \sin(2\pi t/T) \tag{38}$$

where θ_0 and α_0 are amplitudes of pitching and rolling motion, respectively, ϕ is the phase difference between two motions, T is the motion period, and β is the inclination angle of the dragonfly. In this simulation, the kinematic parameters are taken as $\theta_0 = \pi/6$, $\alpha_0 = \pi/4$, $\phi = \pi/2$, $T = 4$, and $\beta = \pi/18$. Additionally, the forewings lead the hindwings by π in phase. Based on the span length of forewing, the Reynolds number used here is $Re = 500$.

Figure 25 illustrates the time histories of force coefficients of dragonfly forewings. Here, the force coefficient is defined as:

$$C_F = F_F/\rho U_\infty^2 S_{\mathrm{W}} \tag{39}$$

where F_F is the force exerted on the dragonfly along the x-, y-, and z-direction and S_{W} is the area of forewing. In the figure, both the force coefficient for the single wing and the sum of two coefficients are included. Because of symmetry to the x–z plane, C_x and C_z at one forewing vary identically with that at the other forewing. As a result, the corresponding sums of C_x and C_z at two wings are increased. On the contrary, the variation of C_y at one forewing varies oppositely with that at the other forewing, and the sum of C_y becomes zero. It can also be found in the figure that all force coefficients reach the peak values twice during each pitching cycle. For the hindwings of the dragonfly, the similar force variations can be found.

It is important for the dragonfly to generate enough lift force in order to keep itself in the air. The lift force generation is mainly related to the pitching motion. Figure 26 shows the evolution of pitching angle and lift coefficient (C_z in this simulation) at one hindwing. Points A and B denote the peak positions of the pitching motion, respectively. Points C and D mean the balance positions where the pitching angle changes its sign. When the hindwing comes to the peak position, the wing velocity in the vertical direction is smallest, and the lift force tends to be zero. When the wing reaches the balance position, the vertical velocity of the wing is the largest, and the lift force also comes to its

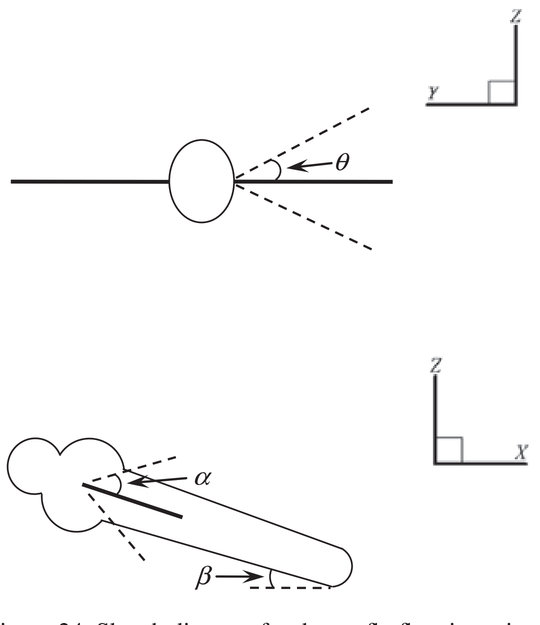


Figure 24. Sketch diagram for dragonfly flapping wing.

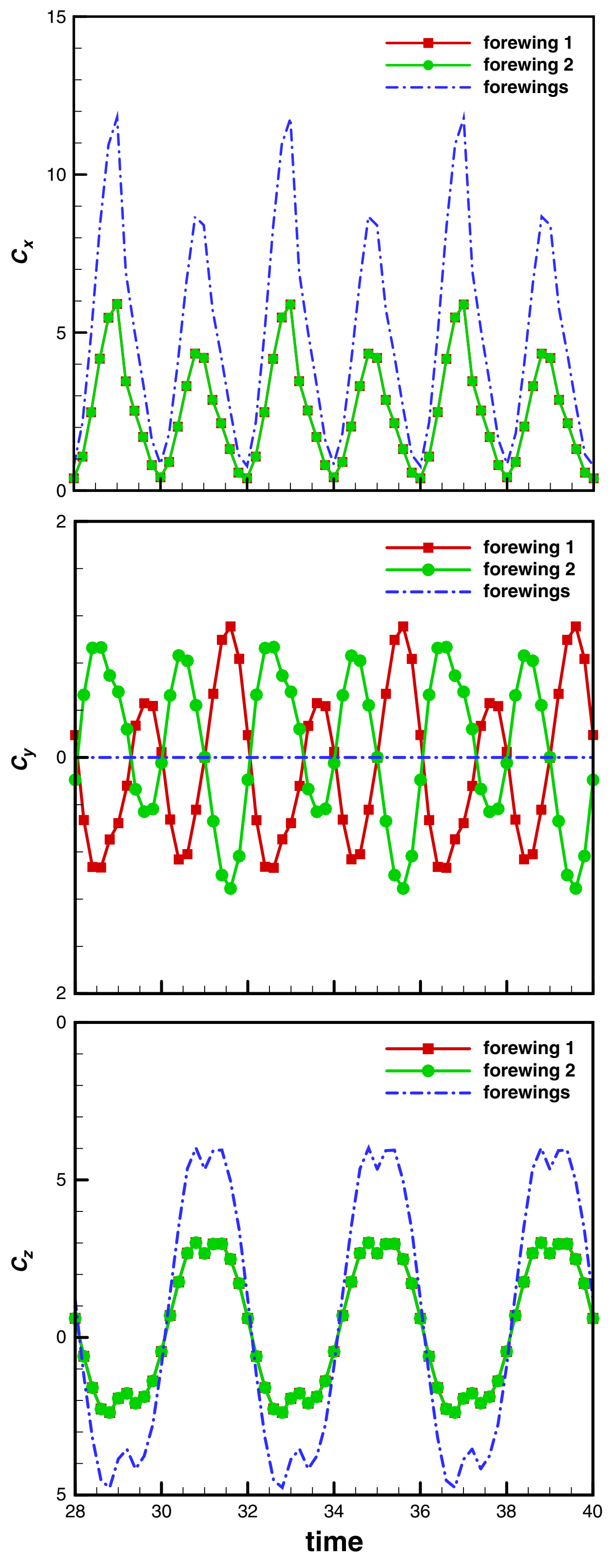


Figure 25. Time histories of x-, y- and z-direction force coefficients for dragonfly flight at $Re = 500$.

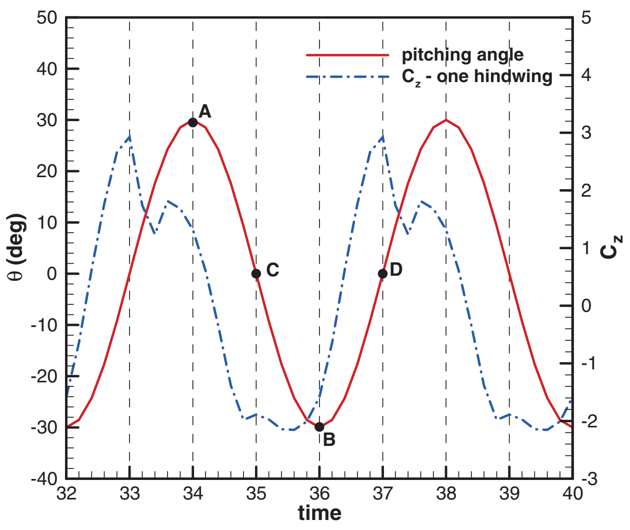


Figure 26. Evolution of pitching angle and lift coefficient at one hindwing.

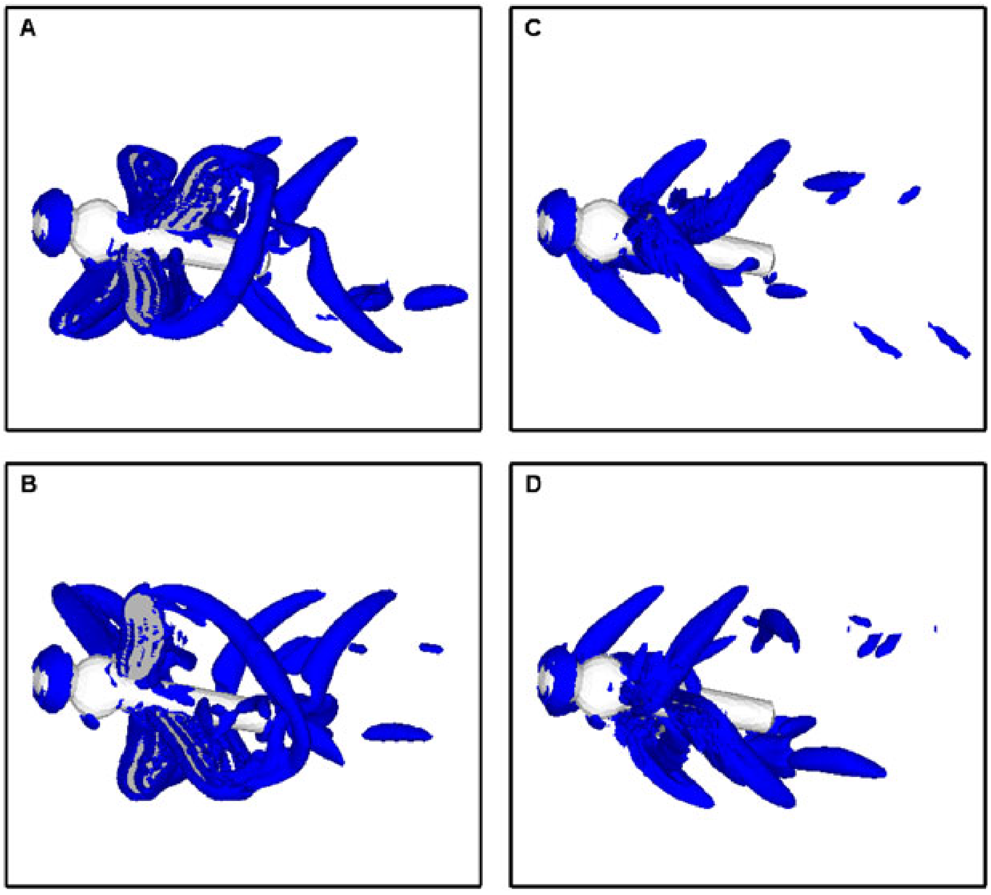


Figure 27. Three-dimensional vortical structures for dragonfly flight at four different stages.

peak value. Figure 27 displays the 3D vortical structures at four corresponding positions. At the pitching peak positions (Points A and B in Figure 26), the wing tip and wake vortices, which are produced during the wing stroke, dominate the flow. At the pitching balance positions (Points C and

D in Figure 26), the dominant features in the flow are the detached leading-edge vortices. It was found that the closer the position to the wing tip, the stronger the vortices.

4. CONCLUSIONS

In this work, a recently developed IB–LBM [28] was applied to simulate 3D flows over moving objects. In this method, because of the enforcement of the non-slip boundary condition, no flow penetration to boundary had appeared, and forces exerted on the objects could be accurately computed. At the same time, an efficient LBM solver, based on the second-order one-dimensional interpolation on the non-uniform Cartesian mesh, was employed. This LBM solver requires much less number of weighting coefficients. It is very efficient for the 3D simulation. In this work, it was also used to obtain the 3D flow field over the moving objects.

To validate the improved IB–LBM solver, four numerical examples were presented. They were the flow over a rotational sphere, the flow over a rotational torus, the fish swimming, and the dragonfly flight. The achieved numerical results and findings were in good agreement with those from previous studies. All the numerical examples demonstrated that the improved IB–LBM is an efficient solver for simulation of 3D flows over moving objects. It has a potential to solve very complicated moving boundary flow problems.

ACKNOWLEDGEMENTS

This work was partially supported by the National Natural Science Foundation of China (10728206).

REFERENCES

1. Saunders BV. A boundary conforming grid generation system for interface tracking. *Computers & Mathematics with Applications* 1995; **29**:1–17.
2. Hu HH. Direct simulation of flows of solid-liquid mixtures. *International Journal of Multiphase Flow* 1996; **22**:335–352.
3. Hu HH, Patankar NA, Zhu MY. Direct numerical simulations of fluid-solid system using arbitrary Lagrangian–Eulerian technique. *Journal of Computational Physics* 2001; **169**:427–462.
4. LeVeque RJ, Li ZL. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM Journal on Numerical Analysis* 1994; **31**:1019–1044.
5. Li ZL, Lai MC. The immersed interface method for the Navier–Stokes equations with singular forces. *Journal of Computational Physics* 2001; **171**:822–842.
6. Calhoun D. A Cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions. *Journal of Computational Physics* 2002; **176**:231–275.
7. Russell D, Wang ZJ. A Cartesian grid method for modeling multiple moving objects in 2D incompressible viscous flow. *Journal of Computational Physics* 2003; **191**:177–205.
8. Xu S, Wang ZJ. An immersed interface method for simulating the interaction of a fluid with moving boundaries. *Journal of Computational Physics* 2006; **216**:454–493.
9. Xu S, Wang ZJ. A 3D immersed interface method for fluid-solid interaction. *Computer Methods in Applied Mechanics and Engineering* 2008; **197**:2068–2086.
10. Udaykumar HS, Mittal R, Shyy W. Computation of solid–liquid phase fronts in the sharp interface limit on fixed grids. *Journal of Computational Physics* 1999; **153**:535–574.
11. Ye T, Mittal R, Udaykumar HS, Shyy W. An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries. *Journal of Computational Physics* 1999; **156**:209–240.
12. Udaykumar HS, Mittal R, Rampunggoon P, Khanna A. A sharp interface Cartesian grid method for simulating flows with complex moving boundaries. *Journal of Computational Physics* 2001; **174**:345–380.
13. Marella S, Krishnan S, Liu H, Udaykumar HS. Sharp interface Cartesian grid method I: an easily implemented technique for 3D moving boundary computations. *Journal of Computational Physics* 2005; **210**:1–31.
14. Peskin CS. Numerical analysis of blood flow in the heart. *Journal of Computational Physics* 1977; **25**:220–252.
15. Roma AM, Peskin CS, Berger MJ. An adaptive version of the immersed boundary method. *Journal of Computational Physics* 1999; **153**:509–534.
16. Tseng Y-H, Ferziger JH. A ghost-cell immersed boundary method for flow in complex geometry. *Journal of Computational Physics* 2003; **192**:593–623.
17. Zhang L, Gerstenberger A, Wang X, Liu WK. Immersed finite element method. *Computer Methods in Applied Mechanics and Engineering* 2004; **193**:2051–2067.
18. Succi S. *The Lattice Boltzmann Equation, For Fluid Dynamics and Beyond*. Oxford University Press: New York, 2001.

19. Feng ZG, Michaelides EE. The immersed boundary–lattice Boltzmann method for solving fluid–particles interaction problems. *Journal of Computational Physics* 2004; **195**:602–628.
20. Feng ZG, Michaelides EE. Proteus: a direct forcing method in the simulations of particulate flows. *Journal of Computational Physics* 2005; **202**:20–51.
21. Peng Y, Shu C, Chew YT, Niu XD, Lu XY. Application of multi-block approach in the immersed boundary–lattice Boltzmann method for viscous fluid flows. *Journal of Computational Physics* 2006; **218**:460–478.
22. Sui Y, Chew Y-T, Roy P, Low HT. A hybrid immersed-boundary and multi-block lattice Boltzmann method for simulating fluid and moving-boundaries interactions. *International Journal for Numerical Methods in Fluids* 2007; **53**:1727–1754.
23. Dupuis A, Chatelain P, Koumoutsakos P. An immersed boundary–lattice Boltzmann method for the simulation of the flow past an impulsively started cylinder. *Journal of Computational Physics* 2008; **227**:4486–4498.
24. Niu XD, Shu C, Chew YT, Peng Y. A momentum exchange-based immersed boundary–lattice Boltzmann method for simulating incompressible viscous flows. *Physics Letters A* 2006; **3**:173–182.
25. Wu J, Shu C. Implicit velocity correction-based immersed boundary–lattice Boltzmann method and its applications. *Journal of Computational Physics* 2009; **228**:1963–1979.
26. Wu J, Shu C, Zhang YH. Simulation of incompressible viscous flows around moving objects by a variant of immersed boundary–lattice Boltzmann method. *International Journal for Numerical Methods in Fluids* 2010; **62**:327–354.
27. Wu J, Shu C. Particulate flow simulation via a boundary-enforced immersed boundary–lattice Boltzmann scheme. *Communications in Computational Physics* 2010; **7**:793–812.
28. Wu J, Shu C. An improved immersed boundary–lattice Boltzmann method for simulating three-dimensional incompressible flows. *Journal of Computational Physics* 2010; **229**:5022–5042.
29. Shu C, Niu XD, Chew YT. Taylor-series expansion and least-squares-based lattice Boltzmann method: two-dimensional formulation and its applications. *Physical Review E* 2002; **65**(036708):1–13.
30. Kim D, Choi H. Laminar flow past a sphere rotating in the streamwise direction. *Journal of Fluid Mechanics* 2002; **461**:365–386.
31. Jeong J, Hussain J. On the identification of a vortex. *Journal of Fluid Mechanics* 1995; **285**:69–94.
32. Sheard GJ, Thompson MC, Hourigan K. From spheres to circular cylinders: the stability and flow structures of bluff ring wakes. *Journal of Fluid Mechanics* 2003; **492**:147–180.
33. Sheard GJ, Hourigan K, Thompson MC. Computations of the drag coefficients for low-Reynolds-number flow past rings. *Journal of Fluid Mechanics* 2005; **526**:257–275.
34. Lighthill MJ. Note on the swimming of slender fish. *Journal of Fluid Mechanics* 1960; **9**:305–317.
35. Muller UK, Van Den Heuvel BLE, Stamhuis EJ, Videler JJ. Fish foot prints: morphology and energetics of the wake behind a continuously swimming mullet Chelon labrosus risso. *The Journal of Experimental Biology* 1997; **200**:2893–2906.
36. Wolfgang MJ, Anderson JM, Grosenbaugh MA, Yue DKP, Triantafyllou MS. Near-body flow dynamics in swimming fish. *The Journal of Experimental Biology* 1999; **202**:2303–2327.
37. Zhu Q, Wolfgang MJ, Yue DKP, Triantafyllou MS. Three-dimensional flow structures and vorticity control in fish-like swimming. *Journal of Fluid Mechanics* 2002; **468**:1–28.
38. Borazjani I, Sotiropoulos F. Numerical investigation of the hydrodynamics of carangiform swimming in the transitional and inertial flow regimes. *The Journal of Experimental Biology* 2008; **211**:1541–1558.
39. Dickinson MH, Lehmann FO, Sane SP. Wing rotation and the aerodynamic basis of insect flight. *Science* 1999; **284**:1954–1960.
40. Birch JM, Dickinson MH. The influence of wing–wake interactions on the production of aerodynamic forces in flapping flight. *The Journal of Experimental Biology* 2003; **206**:2257–2272.
41. Lehmann FO, Sane SP, Dickinson MH. The aerodynamic effects of wing–wing interaction in flapping insect wings. *The Journal of Experimental Biology* 2005; **208**:3075–3092.
42. Mittal R, Dong H, Bozkurttas M, Najjar FM, Vargas A, von Loebbecke A. A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries. *Journal of Computational Physics* 2008; **227**:4825–4852.