Journal of Computational Physics 229 (2010) 5022–5042

Contents lists available at ScienceDirect

Journal of Computational Physics

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

An improved immersed boundary-lattice Boltzmann method for simulating three-dimensional incompressible flows

J. Wu, C. Shu *

Department of Mechanical Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260, Singapore

ARTICLE INFO

Article history:
Received 1 July 2009
Received in revised form 18 January 2010
Accepted 16 March 2010
Available online 21 March 2010

Keywords:
Immersed boundary method
Lattice Boltzmann method
Incompressible viscous flows
Three-dimensional
Boundary condition-enforced
Non-uniform mesh

ABSTRACT

The recently proposed boundary condition-enforced immersed boundary-lattice Boltzmann method (IB-LBM) [14] is improved in this work to simulate three-dimensional incompressible viscous flows. In the conventional IB-LBM, the restoring force is pre-calculated, and the non-slip boundary condition is not enforced as compared to body-fitted solvers. As a result, there is a flow penetration to the solid boundary. This drawback was removed by the new version of IB-LBM [14], in which the restoring force is considered as unknown and is determined in such a way that the non-slip boundary condition is enforced. Since Eulerian points are also defined inside the solid boundary, the computational domain is usually regular and the Cartesian mesh is used. On the other hand, to well capture the boundary layer and in the meantime, to save the computational effort, we often use non-uniform mesh in IB-LBM applications. In our previous two-dimensional simulations [14], the Taylor series expansion and least squares-based lattice Boltzmann method (TLLBM) was used on the non-uniform Cartesian mesh to get the flow field. The final expression of TLLBM is an algebraic formulation with some weighting coefficients. These coefficients could be computed in advance and stored for the following computations. However, this way may become impractical for 3D cases as the memory requirement often exceeds the machine capacity. The other way is to calculate the coefficients at every time step. As a result, extra time is consumed significantly. To overcome this drawback, in this study, we propose a more efficient approach to solve lattice Boltzmann equation on the non-uniform Cartesian mesh. As compared to TLLBM, the proposed approach needs much less computational time and virtual storage. Its good accuracy and efficiency are well demonstrated by its application to simulate the 3D lid-driven cubic cavity flow. To valid the combination of proposed approach with the new version of IBM [14] for 3D flows with curved boundaries, the flows over a sphere and torus are simulated. The obtained numerical results compare very well with available data in the literature.

1. Introduction

Numerical simulation of flows over complex geometries plays an important role in the computational fluid dynamics. Among various numerical solvers, the Cartesian mesh solvers are receiving more and more attention in recent years [1–7] due to their simplicity to simulate flows over complex geometries and moving boundaries. According to

* Corresponding author. Tel.: +65 6516 6476; fax: +65 6779 1459.
E-mail address: mpeshuc@nus.edu.sg (C. Shu).

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.03.024

the treatment of boundary conditions, these solvers can be commonly classified into sharp interface type and diffuse interface type. For a sharp interface method, the boundary is tracked and its thickness is negligible. Cartesian cut-cell method [1,2] and hybrid Cartesian/immersed boundary [3–5] method belong to this type. For a diffuse interface approach, the boundary has a non-zero thickness and it is smeared across some nearby mesh spacing with the use of discrete delta function interpolation. The immersed boundary method (IBM) is the well-known example in this category.

In 1970s, Peskin [6] firstly introduced the immersed boundary method to investigate the blood flow in the human heart. In this method, two independent meshes are employed: the fixed Cartesian mesh which is used to represent the flow field, and the Lagrangian mesh which is used to represent the boundary. Through the discrete delta function interpolation, variables on these two meshes can be related to each other. The key of IBM is that the effect of boundary is described by the restoring force acting on the Eulerian mesh in the vicinity of boundary. Then, Navier–Stokes (N–S) equations with the restoring force are solved over the whole fluid-boundary domain. As compared to body-fitted mesh solvers, IBM decouples the solution of governing equations and implementation of boundary conditions. Due to this feature, IBM has a great potential for simulation of flows with complex geometries and moving boundaries [7].

In traditional IBM applications, the flow field is represented by the solution of incompressible N–S equations. More recently, as an alternative computational approach to the N–S solver, the lattice Boltzmann method (LBM) [8,9] has achieved a great success in a wide range of engineering applications. LBM consists of attractive features: simplicity, easy implementation, algebraic operation and intrinsic parallel nature. Just like the IBM, the standard LBM is usually applied on the Cartesian mesh. Due to this common feature, the combination of IBM and LBM may produce an efficient solver. Some attempt has been made in this aspect. The first effort was made by Feng and Michaelides [10,11] to study the fluid–particle problems. Later, Niu et al. [12] introduced the momentum exchange technique into the IB-LBM, and Peng et al. [13] incorporated the multi-block approach into the IB-LBM.

One common feature of conventional IB-LBM [10–13] is that the boundary effect to the surrounding flow field is through the restoring force, and there is no enforcement of non-slip boundary conditions in the solution process. As a result, some streamlines may penetrate the solid body. To overcome this drawback, Wu and Shu [14] proposed a new version of IB-LBM, in which the restoring force is considered as unknown and is calculated in such a way that the non-slip boundary condition is enforced. As a consequence, the penetration of streamlines to the solid body is avoided, and more accurate numerical results are obtained.

In the work of [14], the solution of flow field is obtained by the lattice Boltzmann method (LBM). It is known that the standard LBM is only applicable on the uniform Cartesian mesh. However, the use of uniform mesh in the IB-LBM applications may not be efficient. This is because around the solid boundary, there is a thin boundary layer. To well capture this boundary layer, the mesh spacing must be very small. So, when the uniform mesh is used, the number of mesh points will be very large in the whole domain, leading to huge computational effort needed. To obtain high resolution near the boundary and in the meantime save the computational time, we need to use non-uniform Cartesian mesh for the LBM solver. Currently, there are a number of approaches [15–20] available in the literature for the application of LBM on the non-uniform mesh. Among them, the Taylor series expansion and least squares-based lattice Boltzmann method (TLLBM) [19,20] is a robust approach, which on one hand, keeps advantage of simple algebraic operation of the standard LBM, and on the other hand, can be well applied to simulate flows with complex geometry and different lattice velocity models. So, this approach is adopted in [14] to combine with the new version of IBM for simulation of various two-dimensional viscous flows.

It is noted that the final form of TLLBM is an algebraic formulation with weighting coefficients. These coefficients are only related to the coordinates of mesh points and lattice velocities. They can be calculated in advance and stored for the following computations. In this way, additional storage is required. The increase of memory requirement for the 2D case is not critical. However, for the 3D case, this way becomes impractical as the memory requirement often exceeds the capacity of the computer. To remove this drawback, we can calculate the coefficients at every time step, but this will considerably increase the computational effort, especially for the 3D case. To efficiently apply IB-LBM to simulate 3D flows, we need to develop a more efficient LBM solver on the non-uniform Cartesian mesh. This motivates the present work.

As can see from this paper, with the feature of non-uniform Cartesian mesh, the distribution function at a position different from the mesh points can be easily calculated by a simple weighted sum constructed by local interpolation along straight mesh lines. Like TLLBM, this approach cannot only be applied on the non-uniform Cartesian mesh but also keeps the advantage of simple algebraic operation of the standard LBM. On the other hand, as compared to TLLBM, much less coefficients are required to be computed and these coefficients can be easily calculated from analytical formulations without involving solution of an equation system. As a consequence, the computational time is greatly reduced. This is confirmed by its application to simulate the three-dimensional lid-driven cavity flow. Numerical results show that the computational efficiency is greatly enhanced as compared with TLLBM. Then the developed LBM solver is incorporated into the new version of IB-LBM [14] for simulation of three-dimensional incompressible viscous flows. Numerical results for flows over a sphere and torus demonstrate that the present solver can effectively simulate three-dimensional flows with curved boundaries.

2. Boundary condition-enforced immersed boundary-lattice Boltzmann method (IB-LBM)

In general, the governing equations for viscous incompressible flows involving immersed boundaries can be written as

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\mathbf{u}\right)+\nabla p=\mu\Delta\mathbf{u}+\mathbf{f} \tag{1}$$

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

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

$$\frac{\partial \mathbf{X}(s,t)}{\partial t}=\mathbf{u}(\mathbf{X}(s,t),t)=\int_{\Omega}\mathbf{u}(\mathbf{x},t)\delta(\mathbf{x}-\mathbf{X}(s,t))\,d\mathbf{x} \tag{4}$$

where $\mathbf{u}$ is the fluid velocity, ρ is the density, p is the pressure, and μ is the dynamic viscosity. $\mathbf{x}$ and $\mathbf{X}$ are Eulerian and Lagrangian coordinates, $\mathbf{f}$ and $\mathbf{F}$ are force density acting on the fluid and boundary, respectively. $\delta(\mathbf{x}-\mathbf{X}(s,t))$ is a Dirac delta function. Eqs. (1) and (2) are the traditional N–S equations for viscous and incompressible fluid with a force density $\mathbf{f}$. Eqs. (3) and (4) give the relationship between the immersed boundary and the fluid, by distributing the boundary force to nearby fluid points and computing the boundary velocity from the fluid velocity.

Based on Chapman–Enskog multi-scale analysis, the lattice Boltzmann equation with forcing term can recover Eqs. (1) and (2). As shown in [14], the corresponding lattice Boltzmann equation can be written as

$$f_\alpha(\mathbf{x}+\mathbf{e}_\alpha\delta t,t+\delta t)-f_\alpha(\mathbf{x},t)=-\frac{1}{\tau}\left(f_\alpha(\mathbf{x},t)-f_\alpha^{eq}(\mathbf{x},t)\right)+F_\alpha\delta t \tag{5}$$

$$F_\alpha=\left(1-\frac{1}{2\tau}\right)w_\alpha\left(\frac{\mathbf{e}_\alpha-\mathbf{u}}{c_s^2}+\frac{\mathbf{e}_\alpha\cdot\mathbf{u}}{c_s^4}\mathbf{e}_\alpha\right)\cdot\mathbf{f} \tag{6}$$

$$\rho\mathbf{u}=\sum_\alpha\mathbf{e}_\alpha f_\alpha+\frac{1}{2}\mathbf{f}\delta t \tag{7}$$

where f_α is the distribution function, f_α^{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 which is distributed from the boundary force, and w_α are the coefficients in the equilibrium distribution function, which depend on the selected lattice velocity model. In our simulation, D3Q15 [21] model as shown in Fig. 1 is used and the velocity set is given by

$$\mathbf{e}_\alpha=\begin{cases}0 & \alpha=0\\ (\pm1,0,0),(0,\pm1,0),(0,0,\pm1) & \alpha=1\text{–}6\\ (\pm1,\pm1,\pm1) & \alpha=7\text{–}14\end{cases} \tag{8}$$

The corresponding equilibrium distribution function [21] is

$$f_\alpha^{eq}(\mathbf{x},t)=\rho w_\alpha\left[1+\frac{\mathbf{e}_\alpha\cdot\mathbf{u}}{c_s^2}+\frac{(\mathbf{e}_\alpha\cdot\mathbf{u})^2-(c_s|\mathbf{u}|)^2}{2c_s^4}\right] \tag{9}$$

where $w_0=2/9,\ w_\alpha=1/9$ for $\alpha=1\text{–}6$ and $w_\alpha=1/72$ for $\alpha=7\text{–}14$. $c_s=c/\sqrt{3}$ is the sound speed of the model, $c=\delta x/\delta t$.

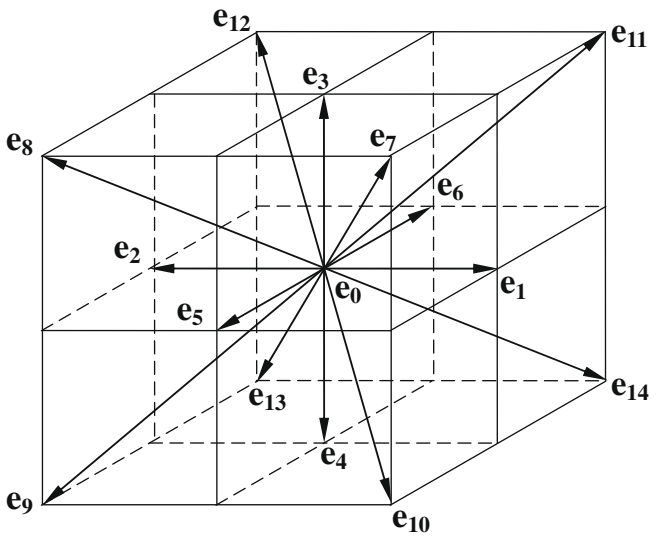

Fig. 1. D3Q15 model.

Eq. (7) indicates that the fluid velocity includes two parts. One is contributed from the density distribution function, and the other is contributed from the force density. By defining the intermediate velocity $\mathbf{u}^* = \sum_\alpha \mathbf{e}_\alpha f_\alpha/\rho$ and velocity correction $\delta\mathbf{u} = \mathbf{f}\delta t/2\rho$, Eq. (7) can be rewritten as

$$\mathbf{u} = \mathbf{u}^* + \delta\mathbf{u} \tag{10}$$

As pointed out in [14], the velocity correction at fluid (Eulerian) point is distributed from the velocity correction at boundary (Lagrangian) points. In the IBM, a set of Lagrangian points $\mathbf{X}_B(s_l,t)$ $(l = 1,2,\ldots,m)$ is used to represent the boundary. Here, we can set an unknown velocity correction vector $\delta\mathbf{u}_B^l$ at every Lagrangian point. The velocity correction $\delta\mathbf{u}$ at the Eulerian point can be obtained by the following Dirac delta function interpolation

$$\delta\mathbf{u}(\mathbf{x},t) = \int_\Gamma \delta\mathbf{u}_B(\mathbf{X}_B,t)\delta(\mathbf{x}-\mathbf{X}_B(s,t))\,ds \tag{11}$$

In the actual implementation, $\delta(\mathbf{x}-\mathbf{X}_B(s,t))$ is smoothly approximated by a continuous kernel distribution D_{ijk} [22]

$$\delta(r) = \begin{cases} \frac{1}{4h}\left(1+\cos\left(\frac{\pi|r|}{2h}\right)\right), & |r| \leqslant 2h \\ 0, & |r| > 2h \end{cases} \tag{12}$$

$$D_{ijk}\left(\mathbf{x}_{ijk}-\mathbf{X}_B^l\right) = \delta\left(x_{ijk}-X_B^l\right)\delta\left(y_{ijk}-Y_B^l\right)\delta\left(z_{ijk}-Z_B^l\right) \tag{13}$$

where h is the mesh spacing of Eulerian mesh. Using Eqs. (13), Eq. (11) can be approximated by

$$\delta\mathbf{u}(\mathbf{x}_{ijk},t) = \sum_l \delta\mathbf{u}_B^l\left(\mathbf{X}_B^l,t\right)D_{ijk}\left(\mathbf{x}_{ijk}-\mathbf{X}_B^l\right)\Delta s_l \quad (l = 1,2,\ldots,m) \tag{14}$$

where Δs_l is the area of the boundary element. In the present simulation, the triangular element is used to discretize the 3D boundary surface.

After obtaining the velocity correction at the Eulerian point, the fluid velocity can be corrected through

$$\mathbf{u}(\mathbf{x}_{ijk},t) = \mathbf{u}^*(\mathbf{x}_{ijk},t) + \delta\mathbf{u}(\mathbf{x}_{ijk},t) \tag{15}$$

where $\mathbf{u}^*(\mathbf{x}_{ijk},t)$ is the intermediate velocity. To satisfy the non-slip boundary condition, the fluid velocity on the boundary point must be equal to the boundary velocity at the same position. The fluid velocity on the boundary point can be obtained by interpolation using the smooth Dirac delta function, and it can be expressed as

$$\mathbf{U}_B^l\left(\mathbf{X}_B^l,t\right) = \sum_{i,j,k}\mathbf{u}(\mathbf{x}_{ijk},t)D_{ijk}\left(\mathbf{x}_{ijk}-\mathbf{X}_B^l\right)\Delta x\Delta y\Delta z \tag{16}$$

When we substitute Eq. (15) into Eq. (16), the following expression can be obtained

$$\mathbf{U}_B^l\left(\mathbf{X}_B^l,t\right) = \sum_{i,j,k}\mathbf{u}^*(\mathbf{x}_{ijk},t)D_{ijk}\left(\mathbf{x}_{ijk}-\mathbf{X}_B^l\right)\Delta x\Delta y\Delta z + \sum_{i,j,k}\sum_l \delta\mathbf{u}_B^l\left(\mathbf{X}_B^l,t\right)D_{ijk}\left(\mathbf{x}_{ijk}-\mathbf{X}_B^l\right)\Delta s_l D_{ijk}\left(\mathbf{x}_{ijk}-\mathbf{X}_B^l\right)\Delta x\Delta y\Delta z \tag{17}$$

If we further set $\delta_{l'j'}^B = D_{ijk}\left(\mathbf{x}_{ijk}-\mathbf{X}_B^l\right)\Delta s_l$ and $\delta_{j'i'} = D_{ijk}\left(\mathbf{x}_{ijk}-\mathbf{X}_B^l\right)\Delta x\Delta y\Delta z$, Eq. (17) can be written in the following matrix form

$$\mathbf{AX} = \mathbf{B} \tag{18}$$

where

$$\mathbf{X} = \left\{\delta\mathbf{u}_B^1,\delta\mathbf{u}_B^2,\ldots,\delta\mathbf{u}_B^m\right\}^T$$

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

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

where m is the number of Lagrangian points, and n is the number of adjacent Eulerian points around the boundary. It can be found that the elements of matrix $\mathbf{A}$ are only related to the coordinates of Lagrangian points and their neighboring Eulerian points. By solving the equation system (18), we can get all the velocity corrections on the Lagrangian points. After that, we

calculate the final corrected fluid velocities by Eqs. (14) and (15). Since the fluid velocity correction is $\delta\mathbf{u} = \mathbf{f}\delta t/2\rho$, we can simply calculate the force density by

$$\mathbf{f}(\mathbf{x}_{ijk}) = 2\rho\delta\mathbf{u}(\mathbf{x}_{ijk})/\delta t \tag{19}$$

In our simulation, the macroscopic variables such as density, intermediate velocity and pressure are calculated by

$$\rho = \sum_{\alpha} f_\alpha, \quad \rho\mathbf{u}^* = \sum_{\alpha} \mathbf{e}_\alpha f_\alpha, \quad P = c_s^2\rho \tag{20}$$

3. An efficient LBM solver on non-uniform Cartesian mesh

As indicated in Section 1, a major contribution of this work is to develop an efficient LBM solver on the non-uniform Cartesian mesh to replace Taylor series expansion- and least square-based LBM (TLLBM) in improving IB-LBM [14] for simulation of three-dimensional flows. To well show the advantages of proposed approach over TLLBM in terms of memory requirement, simplicity and efficiency, we will give a brief description on the TLLBM first before the efficient LBM solver is presented.

3.1. Brief description of TLLBM

As shown in [19,20], TLLBM was developed from the standard LBM, Taylor series expansion and least square optimization. We will take the two-dimensional case as an example to illustrate TLLBM. As shown in Fig. 2, by using the standard LBM, 9 particles initially at mesh points $P_0, P_1, \ldots, P_8$ represented by the symbol "□" will stream to the new positions represented by the symbol "•". For the general case (non-uniform mesh), these new positions may not coincide with the mesh points. To get the distribution function at the mesh point and at the new time level $t + \delta t$, we can apply Taylor series expansion at the new positions and introduce the least square optimization to form an equation system. Its solution gives the distribution function at P_0 and $t + \delta t$ as

$$f_\alpha(P_0, t+\delta t) = \sum_{k=0}^{8} a_{1,k+1}^{\alpha} g_k^{\alpha} \tag{21}$$

where

$$g_k^{\alpha} = f_\alpha(P_k, t) - \left(f_\alpha(P_k, t) - f_\alpha^{eq}(P_k, t)\right)/\tau + F_\alpha\delta t$$

$a_{1,k}^{\alpha}$ are the elements of the first row of the matrix $\mathbf{A}$ given by

$$\mathbf{A} = (\mathbf{S}^T\mathbf{S})^{-1}\mathbf{S}^T$$

$$\mathbf{S} = \begin{pmatrix} 1 & \Delta x_0 & \Delta y_0 & (\Delta x_0)^2/2 & (\Delta y_0)^2/2 & \Delta x_0\Delta y_0 \\ 1 & \Delta x_1 & \Delta y_1 & (\Delta x_1)^2/2 & (\Delta y_1)^2/2 & \Delta x_1\Delta y_1 \\ 1 & - & - & - & - & - \\ 1 & - & - & - & - & - \\ 1 & - & - & - & - & - \\ 1 & \Delta x_8 & \Delta y_8 & (\Delta x_8)^2/2 & (\Delta y_8)^2/2 & \Delta x_8\Delta y_8 \end{pmatrix}_{9\times 6}$$

$$\Delta x_i = x_i + e_{\alpha x}\delta t - x_0, \quad \Delta y_i = y_i + e_{\alpha y}\delta t - y_0 \quad (i = 0, 1, \ldots, 8).$$

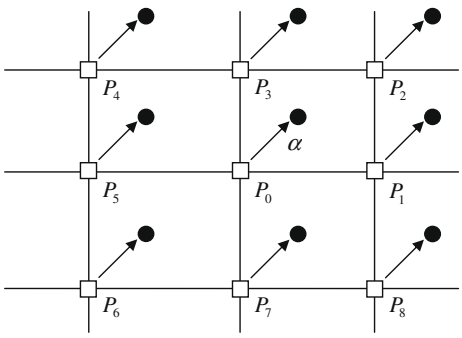

Fig. 2. Schematic diagram of TLLBM (particles are streamed from mesh points).

As can see clearly from Eq. (21), for each lattice direction of the 2D case, we need to compute 9 coefficients which involve matrix multiplication and inversion. The computation will take considerable time as matrix operation is done along each lattice direction. The 3D case is even worse as we have more coefficients and more lattice directions. To save the computational time, we can compute these coefficients once and store them. But this will considerably increase the memory requirement. For example, for the D2Q9 lattice velocity model of the 2D case, we need to store 72 coefficients (8 directions for moving particles $\times$ 9 points) at every mesh point, while for the D3Q15 model of the 3D case, we have to store 210 coefficients (14 directions for moving particles $\times$ 15 points) at every mesh point. It seems that either computing the coefficients at every time step or storing them in the memory is not an efficient way in application of TLLBM to the 3D case. This difficulty can be overcome by the following approach.

3.2. Efficient LBM solver on non-uniform Cartesian mesh

Lattice Boltzmann equation (5) can be rewritten as

$$f_\alpha(\mathbf{x},t) = g_\alpha(\mathbf{x}-\mathbf{e}_\alpha\delta t, t-\delta t) \tag{22}$$

$$g_\alpha(\mathbf{x}-\mathbf{e}_\alpha\delta t, t-\delta t) = f_\alpha(\mathbf{x}-\mathbf{e}_\alpha\delta t, t-\delta t) + \frac{1}{\tau}\left(f_\alpha^{eq}(\mathbf{x}-\mathbf{e}_\alpha\delta t, t-\delta t) - f_\alpha(\mathbf{x}-\mathbf{e}_\alpha\delta t, t-\delta t)\right) + F_\alpha\delta t \tag{23}$$

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. Hence, the value of g_α at the position $\mathbf{x}-\mathbf{e}_\alpha\delta t$ has to be obtained first before the streaming process starts. For the general case, this may not be an easy job. However, as mentioned above, IB-LBM is usually applied on the non-uniform Cartesian mesh. For this case, we can easily evaluate g_α at $\mathbf{x}-\mathbf{e}_\alpha\delta t$ by using the second order local interpolation. Take the two-dimensional case and D2Q9 model as an example. As shown in Fig. 3, 8 moving particles at "□" positions, which may not be the mesh points, will stream to the mesh point "•". Since the non-uniform Cartesian mesh (mesh lines are straight lines) is used, evaluation of g_α ($\alpha = 1$—8) at "□" positions is simple and straightforward. Note that for all the cases, g_α is evaluated at the time level t. In the following, for simplicity of derivation, t is taken out from g_α. We firstly consider the evaluation of g_1. As seen in Fig. 3, g_1 is on the horizontal mesh line of $y = y_j$. Using distribution function values at three mesh points (x_{i-1}, y_j), (x_i, y_j), and (x_{i+1}, y_j), g_1 can be obtained by the second order interpolation as

$$g_1(x,y_j) = a_1 g_1(x_{i-1},y_j) + a_2 g_1(x_i,y_j) + a_3 g_1(x_{i+1},y_j) \tag{24}$$

Here a_1, a_2, a_3 are Lagrange interpolation coefficients, which can be expressed as

$$\begin{aligned} a_1 &= \frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})} \\ a_2 &= \frac{(x-x_{i-1})(x-x_{i+1})}{(x_i-x_{i-1})(x_i-x_{i+1})} \\ a_3 &= \frac{(x-x_{i-1})(x-x_i)}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)} \end{aligned} \tag{25}$$

If we define

$$\Delta x_1 = x_{i-1} - x_i, \quad \Delta x_2 = x_{i+1} - x_i \tag{26}$$

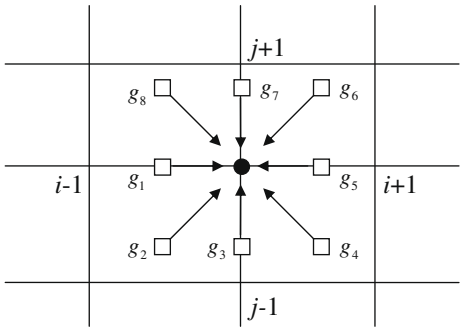

Fig. 3. Sketch of local interpolation scheme (particles are streamed from "□" points).

and use $x = x_i - e_{1x}\delta t$, Eq. (25) can be rewritten as

$$\begin{aligned} a_1 &= \frac{e_{1x}\delta t(e_{1x}\delta t + \Delta x_2)}{\Delta x_1(\Delta x_1 - \Delta x_2)} \\ a_2 &= \frac{(e_{1x}\delta t + \Delta x_1)(e_{1x}\delta t + \Delta x_2)}{\Delta x_1 \Delta x_2} \\ a_3 &= \frac{e_{1x}\delta t(e_{1x}\delta t + \Delta x_1)}{\Delta x_2(\Delta x_2 - \Delta x_1)} \end{aligned} \tag{27}$$

Here e_{1x} is the x-component of lattice velocity along the direction 1, which is 1 in the D2Q9 model. Similarly, g_3 on the vertical mesh line of $x = x_i$ can be evaluated by

$$g_3(x_i, y) = b_1 g_3(x_i, y_{j-1}) + b_2 g_3(x_i, y_j) + b_3 g_3(x_i, y_{j+1}) \tag{28}$$

where $b_1,\ b_2,\ b_3$ are

$$\begin{aligned} b_1 &= \frac{e_{3y}\delta t(e_{3y}\delta t + \Delta y_2)}{\Delta y_1(\Delta y_1 - \Delta y_2)} \\ b_2 &= \frac{(e_{3y}\delta t + \Delta y_1)(e_{3y}\delta t + \Delta y_2)}{\Delta y_1 \Delta y_2} \\ b_3 &= \frac{e_{3y}\delta t(e_{3y}\delta t + \Delta y_1)}{\Delta y_2(\Delta y_2 - \Delta y_1)} \end{aligned} \tag{29}$$

with

$$\Delta y_1 = y_{j-1} - y_j, \quad \Delta y_2 = y_{j+1} - y_j \tag{30}$$

and e_{3y} is the y-component of lattice velocity along the direction 3, which is 1 in the D2Q9 model. Eq. (24) is actually the one-dimensional interpolation along the horizontal mesh line of $y = y_j$, while Eq. (28) is the one-dimensional interpolation along the vertical mesh line of $x = x_i$. They can also be applied to evaluate g_5 and g_7 when e_{1x} is replaced by e_{5x} and e_{3y} is replaced by e_{7y} (both e_{5x} and e_{7y} take the value of -1 in the D2Q9 model). For other positions which are neither on the horizontal mesh line nor on the vertical mesh line, we can combine Eqs. (24) and (28) to evaluate the distribution function. Suppose that we need to evaluate $g_\alpha(x, y)$ for a general case. At first, we can apply Eq. (24) to get the intermediate distribution functions $g_\alpha(x, y_{j-1}),\ g_\alpha(x, y_j)$, and $g_\alpha(x, y_{j+1})$ at three locations $(x, y_{j-1}),\ (x, y_j)$, and (x, y_{j+1}). The expressions can be written as

$$\begin{aligned} g_\alpha(x, y_{j-1}) &= a_{1\alpha} g_\alpha(x_{i-1}, y_{j-1}) + a_{2\alpha} g_\alpha(x_i, y_{j-1}) + a_{3\alpha} g_\alpha(x_{i+1}, y_{j-1}) \\ g_\alpha(x, y_j) &= a_{1\alpha} g_\alpha(x_{i-1}, y_j) + a_{2\alpha} g_\alpha(x_i, y_j) + a_{3\alpha} g_\alpha(x_{i+1}, y_j) \\ g_\alpha(x, y_{j+1}) &= a_{1\alpha} g_\alpha(x_{i-1}, y_{j+1}) + a_{2\alpha} g_\alpha(x_i, y_{j+1}) + a_{3\alpha} g_\alpha(x_{i+1}, y_{j+1}) \end{aligned} \tag{31}$$

with

$$\begin{aligned} a_{1\alpha} &= \frac{e_{\alpha x}\delta t(e_{\alpha x}\delta t + \Delta x_2)}{\Delta x_1(\Delta x_1 - \Delta x_2)} \\ a_{2\alpha} &= \frac{(e_{\alpha x}\delta t + \Delta x_1)(e_{\alpha x}\delta t + \Delta x_2)}{\Delta x_1 \Delta x_2} \\ a_{3\alpha} &= \frac{e_{\alpha x}\delta t(e_{\alpha x}\delta t + \Delta x_1)}{\Delta x_2(\Delta x_2 - \Delta x_1)} \end{aligned} \tag{32}$$

Then by using these intermediate values and Eq. (28), $g_\alpha(x, y)$ can be easily calculated by

$$g_\alpha(x, y) = b_{1\alpha} g_\alpha(x, y_{j-1}) + b_{2\alpha} g_\alpha(x, y_j) + b_{3\alpha} g_\alpha(x, y_{j+1}) \tag{33}$$

Here,

$$\begin{aligned} b_{1\alpha} &= \frac{e_{\alpha y}\delta t(e_{\alpha y}\delta t + \Delta y_2)}{\Delta y_1(\Delta y_1 - \Delta y_2)} \\ b_{2\alpha} &= \frac{(e_{\alpha y}\delta t + \Delta y_1)(e_{\alpha y}\delta t + \Delta y_2)}{\Delta y_1 \Delta y_2} \\ b_{3\alpha} &= \frac{e_{\alpha y}\delta t(e_{\alpha y}\delta t + \Delta y_1)}{\Delta y_2(\Delta y_2 - \Delta y_1)} \end{aligned} \tag{34}$$

By setting $x = x_i - e_{\alpha x}\delta t,\ y = y_j - e_{\alpha y}\delta t$, Eq. (33) can be put in the following matrix form,

$$g_\alpha(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t) = \mathbf{BGA} \tag{35}$$

where matrices **A**, **B** and **G** are

$$\mathbf{A} = \{a_{1\alpha}, a_{2\alpha}, a_{3\alpha}\}^T, \quad \mathbf{B} = \{b_{1\alpha}, b_{2\alpha}, b_{3\alpha}\} \tag{36}$$

$$\mathbf{G} = \begin{pmatrix} g_\alpha(x_{i-1}, y_{j-1}) & g_\alpha(x_i, y_{j-1}) & g_\alpha(x_{i+1}, y_{j-1}) \\ g_\alpha(x_{i-1}, y_j) & g_\alpha(x_i, y_j) & g_\alpha(x_{i+1}, y_j) \\ g_\alpha(x_{i-1}, y_{j+1}) & g_\alpha(x_i, y_{j+1}) & g_\alpha(x_{i+1}, y_{j+1}) \end{pmatrix} \tag{37}$$

As can be seen from Eqs. (32) and (34), for the given non-uniform mesh, the coefficients depend on the lattice velocity. In particular, when $e_{\alpha x} = 0,\ e_{\alpha y} = 0$, then $a_{1\alpha} = a_{3\alpha} = 0,\ a_{2\alpha} = 1,\ b_{1\alpha} = b_{3\alpha} = 0,\ b_{2\alpha} = 1$. In fact, for this case, the particle is static and there is no streaming process. Apart from the static particle, in the D2Q9 lattice model, both $e_{\alpha x}$ and $e_{\alpha y}$ take either 1 or -1. This means that we only have two sets of coefficient matrices **A** and **B**. So, overall, we have 12 coefficients (6 in the x-direction and 6 in the y-direction) to store. Obviously, as compared to the TLLBM where 72 coefficients are needed to store, the memory requirement of present approach is greatly reduced. In addition, the calculation of coefficients in the present work also takes much less time as compared with TLLBM. In this work, they are simply calculated by algebraic formulations (32) and (34), while in TLLBM, they are obtained by solving an algebraic equation system (involving matrix inversion). On the other hand, like TLLBM, the application of present approach keeps the advantage of the standard LBM, that is, simple algebraic operation to update distribution functions.

The above procedure to evaluate $g_\alpha(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t)$ for the 2D case can be directly extended to the 3D case. For the 3D case, we can apply Eq. (35) to calculate the intermediate distribution functions $g_\alpha(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t, z_{k-1})$, $g_\alpha(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t, z_k)$, and $g_\alpha(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t, z_{k+1})$ at three planes of $z = z_{k-1},\ z = z_k$, and $z = z_{k+1}$ first. The corresponding formulations are

$$\begin{aligned} g_\alpha(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t, z_{k-1}) &= \mathbf{B}\mathbf{G}_1\mathbf{A} \\ g_\alpha(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t, z_k) &= \mathbf{B}\mathbf{G}_2\mathbf{A} \\ g_\alpha(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t, z_{k+1}) &= \mathbf{B}\mathbf{G}_3\mathbf{A} \end{aligned} \tag{38}$$

where matrices **A** and **B** are defined in Eq. (36). The matrices $\mathbf{G}_1,\ \mathbf{G}_2$ and $\mathbf{G}_3$ are

$$\begin{aligned} \mathbf{G}_1 &= \begin{pmatrix} g_\alpha(x_{i-1}, y_{j-1}, z_{k-1}) & g_\alpha(x_i, y_{j-1}, z_{k-1}) & g_\alpha(x_{i+1}, y_{j-1}, z_{k-1}) \\ g_\alpha(x_{i-1}, y_j, z_{k-1}) & g_\alpha(x_i, y_j, z_{k-1}) & g_\alpha(x_{i+1}, y_j, z_{k-1}) \\ g_\alpha(x_{i-1}, y_{j+1}, z_{k-1}) & g_\alpha(x_i, y_{j+1}, z_{k-1}) & g_\alpha(x_{i+1}, y_{j+1}, z_{k-1}) \end{pmatrix} \\ \mathbf{G}_2 &= \begin{pmatrix} g_\alpha(x_{i-1}, y_{j-1}, z_k) & g_\alpha(x_i, y_{j-1}, z_k) & g_\alpha(x_{i+1}, y_{j-1}, z_k) \\ g_\alpha(x_{i-1}, y_j, z_k) & g_\alpha(x_i, y_j, z_k) & g_\alpha(x_{i+1}, y_j, z_k) \\ g_\alpha(x_{i-1}, y_{j+1}, z_k) & g_\alpha(x_i, y_{j+1}, z_k) & g_\alpha(x_{i+1}, y_{j+1}, z_k) \end{pmatrix} \\ \mathbf{G}_3 &= \begin{pmatrix} g_\alpha(x_{i-1}, y_{j-1}, z_{k+1}) & g_\alpha(x_i, y_{j-1}, z_{k+1}) & g_\alpha(x_{i+1}, y_{j-1}, z_{k+1}) \\ g_\alpha(x_{i-1}, y_j, z_{k+1}) & g_\alpha(x_i, y_j, z_{k+1}) & g_\alpha(x_{i+1}, y_j, z_{k+1}) \\ g_\alpha(x_{i-1}, y_{j+1}, z_{k+1}) & g_\alpha(x_i, y_{j+1}, z_{k+1}) & g_\alpha(x_{i+1}, y_{j+1}, z_{k+1}) \end{pmatrix} \end{aligned} \tag{39}$$

Finally, $g_\alpha(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t, z_k - e_{\alpha z}\delta t)$ can be calculated by the one-dimensional interpolation along the z direction as

$$\begin{aligned} g_\alpha(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t, z_k - e_{\alpha z}\delta t) &= c_{1\alpha} g_\alpha(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t, z_{k-1}) + c_{2\alpha} g_\alpha(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t, z_k) \\ &\quad + c_{3\alpha} g_\alpha(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t, z_{k+1}) = \mathbf{C}\mathbf{G}_\mathbf{t} \end{aligned} \tag{40}$$

where $\mathbf{G}_\mathbf{t} = \{\mathbf{B}\mathbf{G}_1\mathbf{A}, \mathbf{B}\mathbf{G}_2\mathbf{A}, \mathbf{B}\mathbf{G}_3\mathbf{A}\}^T,\ \mathbf{C} = \{c_{1\alpha}, c_{2\alpha}, c_{3\alpha}\}$ with

$$\begin{aligned} c_{1\alpha} &= \frac{e_{\alpha z}\delta t(e_{\alpha z}\delta t + \Delta z_2)}{\Delta z_1(\Delta z_1 - \Delta z_2)} \\ c_{2\alpha} &= \frac{(e_{\alpha z}\delta t + \Delta z_1)(e_{\alpha z}\delta t + \Delta z_2)}{\Delta z_1 \Delta z_2} \\ c_{3\alpha} &= \frac{e_{\alpha z}\delta t(e_{\alpha z}\delta t + \Delta z_1)}{\Delta z_2(\Delta z_2 - \Delta z_1)} \end{aligned} \tag{41}$$

Here Δz_1 and Δz_2 are

$$\Delta z_1 = z_{k-1} - z_k, \quad \Delta z_2 = z_{k+1} - z_k \tag{42}$$

After $g_\alpha(x_i - e_{\alpha x}\delta t, y_j - e_{\alpha y}\delta t, z_k - e_{\alpha z}\delta t)$ is calculated, it is very easy to get $f_\alpha(x_i, y_j, z_k)$ by using Eq. (22). Note that in the present approach, the second order interpolation is adopted.

We can see from Eq. (40) that as compared to the 2D case, we only have additional coefficients $c_{1\alpha},\ c_{2\alpha},\ c_{3\alpha}$ for the 3D case. Their values can be calculated by analytical expression (41). When D3Q15 lattice model is adopted, $e_{\alpha x},\ e_{\alpha y}$, and $e_{\alpha z}$ also take either 1 or -1. So, in each direction, we need to store 6 coefficients only. Overall, we need to store 18 coefficients for each mesh point. As shown in Section 3.1, TLLBM needs to store 210 coefficients for this case. The memory saving of present approach is

obvious. If the coefficients are computed every time step, the increase of computational effort is not much since they are given by simple algebraic formulations (32), (34) and (41) without involving the solution of an algebraic equation system.

On the other hand, we have to indicate that although the present approach is more efficient than TLLBM. Its application is not as general as TLLBM. TLLBM can be applied to any point distribution. In contrast, the present approach is only applicable on the structured non-uniform Cartesian mesh. Fortunately, IB-LBM is applied on such mesh. In this sense, we can say that the present approach is specifically developed for IB-LBM to improve its computational efficiency.

When the present approach is incorporated into IB-LBM, its basic solution procedure for simulating a three-dimensional flow can be summarized as follows,

(1) Set initial conditions, compute the elements of matrix **A** in Eq. (18) and get $\mathbf{A}^{-1}$.
(2) Use Eq. (5) to obtain the density distribution function at time level $t = t_n$ (initially setting $F_\alpha = 0$). The post-collision state of distribution function in Eq. (5) is computed using our proposed LBM solver on the non-uniform Cartesian mesh, i.e. Eq. (40). Compute the macroscopic variables using Eq. (20).
(3) Solve equation system (18) to get the velocity corrections at all boundary points and use Eq. (14) to get the fluid velocity corrections.
(4) Correct the fluid velocity using Eq. (15) and obtain the force density using Eq. (19).
(5) Compute the equilibrium distribution function using Eq. (9).
(6) Repeat Steps 2–5 until converged solution is reached.

3.3. Accuracy analysis of present LBM

To demonstrate the second order accuracy of proposed LBM, a mathematical analysis is shown in this section. For simplicity, the one-dimensional model is selected to illustrate the analysis. According to Eq. (22), $f_\alpha(x_i, t)$ is given by

$$f_\alpha(x_i, t) = g_\alpha(x_i - e_\alpha \delta t, t - \delta t) \tag{43}$$

To apply the above equation, we notice that the position $x_i - e_\alpha \delta t$ may not be the mesh point for the non-uniform Cartesian mesh. Thus, we have to use the local interpolation. According to Eq. (24), $f_\alpha(x_i, t)$ in our proposed LBM is evaluated by

$$f_\alpha(x_i, t) = a_{1\alpha} g_\alpha(x_{i-1}, t - \delta t) + a_{2\alpha} g_\alpha(x_i, t - \delta t) + a_{3\alpha} g_\alpha(x_{i+1}, t - \delta t) \tag{44}$$

where $x_{i-1},\ x_i$, and x_{i+1} are coordinates of the three mesh points, and

$$g_\alpha(x_k, t - \delta t) = f_\alpha(x_k, t - \delta t) + \frac{1}{\tau}\left(f_\alpha^{eq}(x_k, t - \delta t) - f_\alpha(x_k, t - \delta t)\right) \quad (k = i-1, i, i+1)$$
$$a_{1\alpha} = \frac{e_\alpha \delta t (e_\alpha \delta t + \Delta x_2)}{\Delta x_1 (\Delta x_1 - \Delta x_2)}, \quad a_{2\alpha} = \frac{(e_\alpha \delta t + \Delta x_1)(e_\alpha \delta t + \Delta x_2)}{\Delta x_1 \Delta x_2}, \quad a_{3\alpha} = \frac{e_\alpha \delta t (e_\alpha \delta t + \Delta x_1)}{\Delta x_2 (\Delta x_2 - \Delta x_1)}$$
$$\Delta x_1 = x_{i-1} - x_i, \quad \Delta x_2 = x_{i+1} - x_i$$

In the following, we will show that as compared with the original formulation (43), Eq. (44) has the second order of accuracy. Since $g_\alpha(x_{i-1}, t - \delta t),\ g_\alpha(x_i, t - \delta t),\ g_\alpha(x_{i+1}, t - \delta t)$ and $g_\alpha(x_i - e_\alpha \delta t, t - \delta t)$ are all evaluated at the time level $t - \delta t$, for simplicity of following derivation, $t - \delta t$ is taken out from g_α. Using Taylor series expansion, $g_\alpha(x_{i-1}),\ g_\alpha(x_i),\ g_\alpha(x_{i+1})$ can be approximated by $g_\alpha(x_i - e_\alpha \delta t)$ and its derivatives as

$$g_\alpha(x_{i-1}) = g_\alpha(x_i - e_\alpha \delta t) + (\Delta x_1 + e_\alpha \delta t)\frac{\partial g_\alpha}{\partial x} + \frac{1}{2}(\Delta x_1 + e_\alpha \delta t)^2 \frac{\partial^2 g_\alpha}{\partial x^2} + O(\delta t^3) \tag{45a}$$

$$g_\alpha(x_i) = g_\alpha(x_i - e_\alpha \delta t) + e_\alpha \delta t \frac{\partial g_\alpha}{\partial x} + \frac{1}{2}(e_\alpha \delta t)^2 \frac{\partial^2 g_\alpha}{\partial x^2} + O(\delta t^3) \tag{45b}$$

$$g_\alpha(x_{i+1}) = g_\alpha(x_i - e_\alpha \delta t) + (\Delta x_2 + e_\alpha \delta t)\frac{\partial g_\alpha}{\partial x} + \frac{1}{2}(\Delta x_2 + e_\alpha \delta t)^2 \frac{\partial^2 g_\alpha}{\partial x^2} + O(\delta t^3) \tag{45c}$$

Note that in the above equation, the spatial derivatives on the right hand side are evaluated at $(x_i - e_\alpha \delta t)$. Substituting Eq. (45) into Eq. (44) gives

$$f_\alpha(x_i, t) = s_1 g_\alpha(x_i - e_\alpha \delta t, t - \delta t) + s_2 \frac{\partial g_\alpha}{\partial x} + s_3 \frac{\partial^2 g_\alpha}{\partial x^2} + O(\delta t^3) \tag{46}$$

where

$$s_1 = a_{1\alpha} + a_{2\alpha} + a_{3\alpha}$$
$$s_2 = a_{1\alpha}(\Delta x_1 + e_\alpha \delta t) + a_{2\alpha} e_\alpha \delta t + a_{3\alpha}(\Delta x_2 + e_\alpha \delta t)$$
$$s_3 = \frac{1}{2} a_{1\alpha}(\Delta x_1 + e_\alpha \delta t)^2 + \frac{1}{2} a_{2\alpha}(e_\alpha \delta t)^2 + \frac{1}{2} a_{3\alpha}(\Delta x_2 + e_\alpha \delta t)^2$$

With the expressions of $a_{1\alpha},\ a_{2\alpha},\ a_{3\alpha}$, it is easy to show that $s_1 = 1,\ s_2 = 0,\ s_3 = 0$. This means that Eq. (44) can be accurate to Eq. (43) up to $(\delta t)^2$ term (truncation error is in the order of $(\delta t)^3$). Therefore, we can conclude that our proposed LBM has the second order of accuracy as compared to the standard LBM.

4. Numerical results and discussion

4.1. Flow in 3D lid-driven cavity

To validate the proposed efficient LBM solver on the non-uniform Cartesian mesh, the simulation of 3D lid-driven cubic cavity flow is carried out and its efficiency is demonstrated by comparing the CPU time consumed in this method with that in TLLBM. The Reynolds number is defined as $\mathrm{Re} = U_\infty L/\upsilon$, which is based on the lid velocity and length of cavity. Here, the cases of Re = 100, 400 and 1000 are considered. Three non-uniform meshes of $61 \times 61 \times 61$, $81 \times 81 \times 81$, and $121 \times 121 \times 121$ with fine mesh near the cavity boundaries and coarse mesh near the center are used. The mesh point is generated by

$$x_i = 0.5(1 - \eta \tan^{-1}((1 - \kappa_i) \cdot \tan(1/\eta))) \quad (i = 1, 2, 3) \tag{47}$$

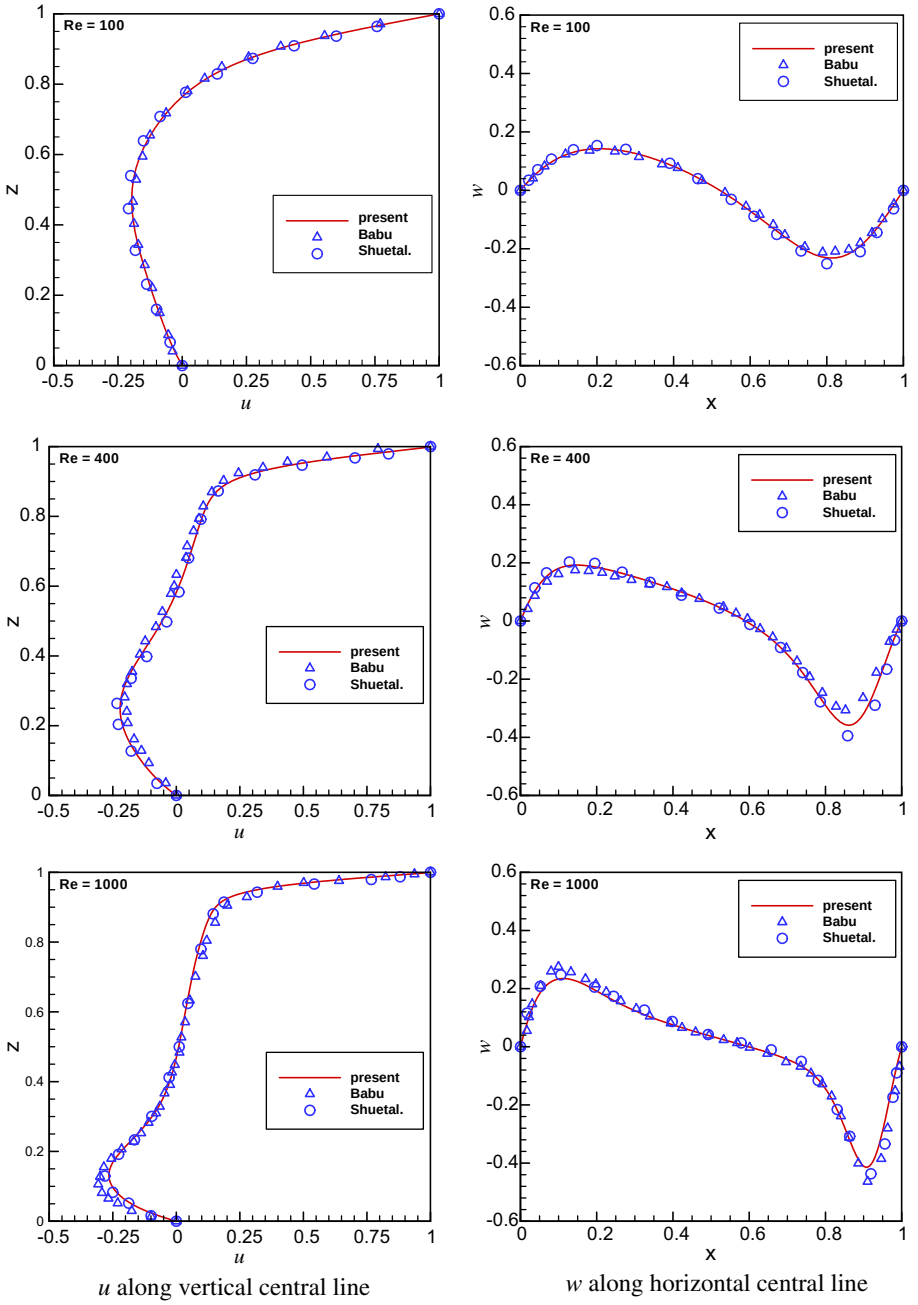

Fig. 4. Comparison of velocity profiles on the plane of $y = 0.5$.

where x_1, x_2, and x_3 are mesh coordinates along the x-, y-, and z-direction, respectively; $\kappa_i = (I_i - 1)/((Im_i - 1)2)$, I_i and Im_i are the mesh point index and mesh size along a spatial direction, respectively; η is the parameter to control the mesh stretching, which is selected as 0.85 in this study.

In current simulation, the fluid density is taken as $\rho = 1.0$ and the lid velocity is $U_\infty = 0.1$. Initially, the density inside the cavity is constant and velocity is zero. The lid on the top moves along the x-direction. The equilibrium distribution functions are used to implement moving boundary condition and bounce-back schemes are applied on the stationary walls. The convergence criterion for simulations is set as

$$\sum \left(\left| \mathbf{u}_{i,j,k}^{n+1} \right| - \left| \mathbf{u}_{i,j,k}^{n} \right| \right) \leqslant 10^{-8}$$

Fig. 4 plots the velocity profiles of x-direction component u along the vertical centerline and z-direction component w along the horizontal centerline on the plane of $y = 0.5$ for three Reynolds numbers. For comparison, the results of Babu and Korpela [23] obtained by solving incompressible N–S equations and those of Shu et al. [24] using TLLBM are also

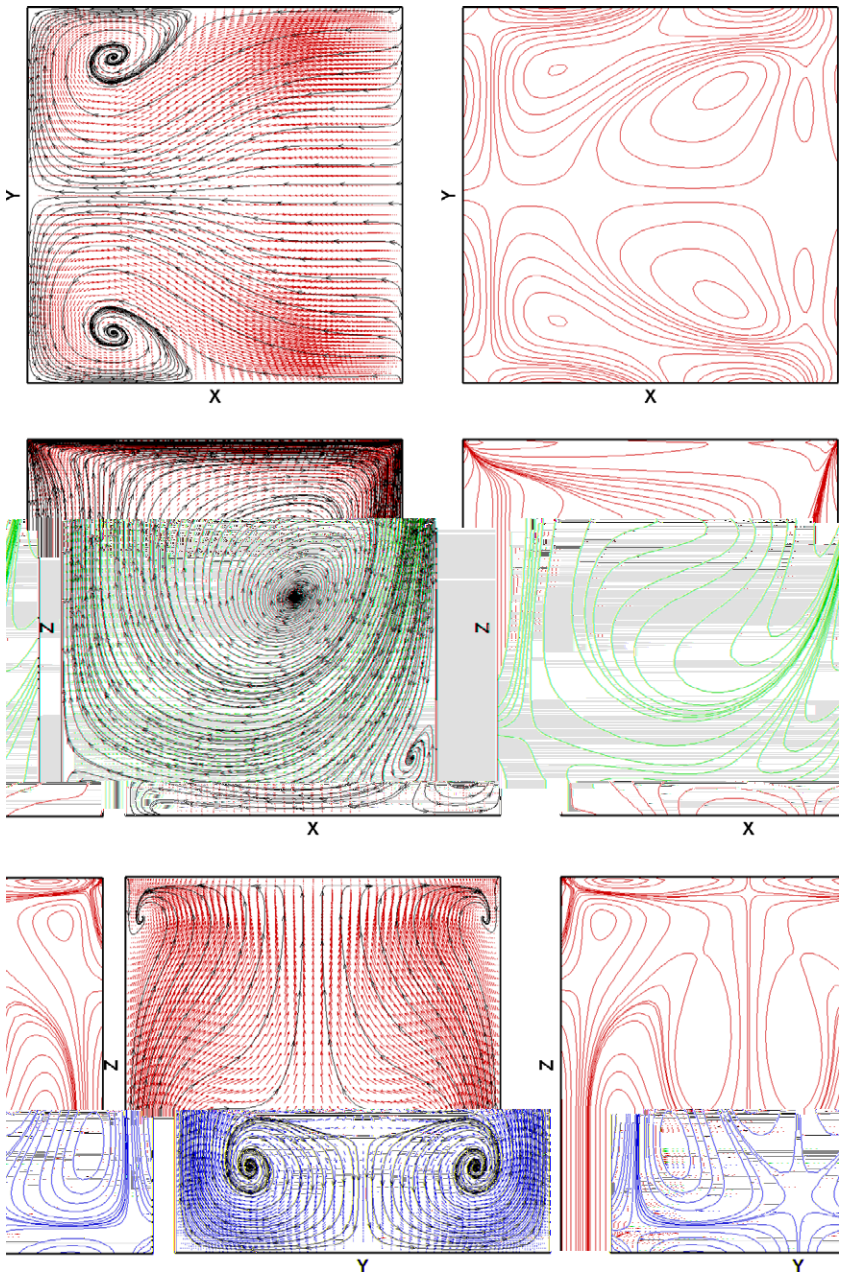

Fig. 5. Streamtraces and vorticity contours on three mid-planes for Re = 400.

included. It is clear that the present results compare very well with previous numerical results. Fig. 5 shows the streamtraces and vorticity contours on three orthogonal mid-planes for Re = 400. These plots also agree well with those in the previous study.

In TLLBM, the required coefficients can be computed in advance. For D3Q15, we need to store at least 14 coefficients for every mesh point and every lattice velocity direction [24]. When the mesh size is increased, it becomes unacceptable to store so many coefficients due to the limitation of virtual memory. Another choice is to calculate coefficients during every time step. Hence, a great deal of extra time is consumed due to tedious matrix operation. In our proposed LBM solver, the coefficients are simply calculated by analytical formulations (32), (34) and (41). As compared to TLLBM, very less extra time is needed. Take the non-uniform mesh of $61 \times 61 \times 61$ as an example. The CPU time consumed by the present approach and

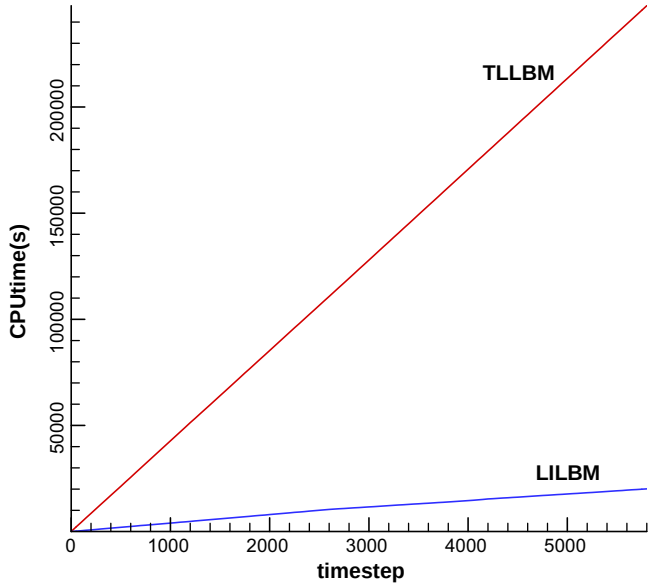

Fig. 6. Comparison of CPU time.

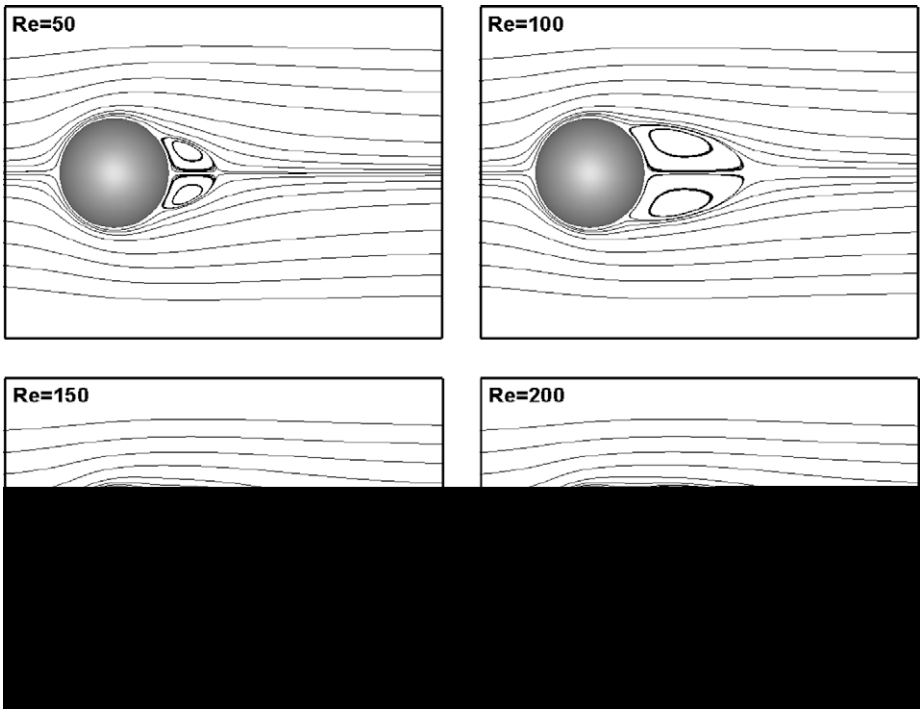

Fig. 7. Streamlines at the x–y plane for steady axisymmetric flows.

TLLBM is compared, and the results are displayed in Fig. 6, which shows the CPU time against computational time step. It can be found that the relationship between CPU time and time step is linear. The slope for TLLBM and present approach is 42.7 and 3.45, respectively. This implicitly means that the present solver only takes about 3.45/42.7 = 8% of computational time of TLLBM. The high efficiency of the proposed LBM solver is obvious.

4.2. Flow over a stationary sphere

To simulate 3D flow with immersed boundaries, we combine the proposed LBM solver with boundary condition-enforced IB-LBM [14]. The example of flow over a sphere is selected to validate the improved IB-LBM. The behavior of this three-dimensional flow with varying Reynolds numbers has been extensively studied. Here the low Reynolds number ($\mathrm{Re} \leqslant 300$) laminar flows are simulated. The Reynolds number is defined as

$$\mathrm{Re} = \frac{U_\infty D}{\upsilon} \tag{48}$$

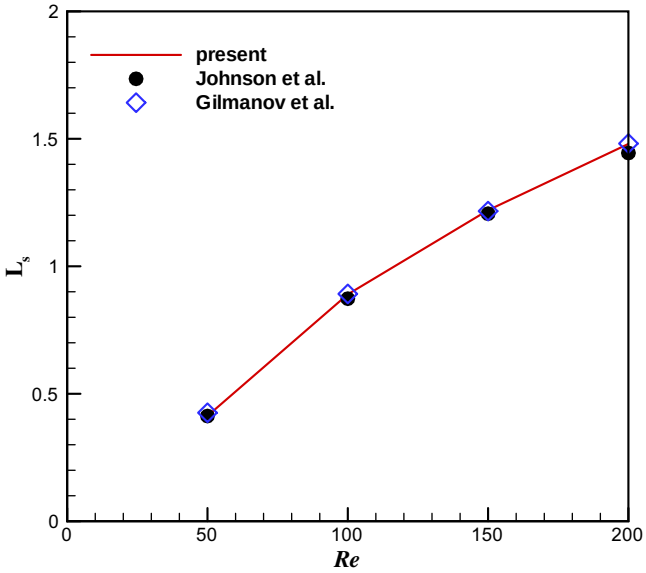

Fig. 8. Comparison of recirculation length L_s.

Table 1
Comparison of drag coefficients at Re = 100 and 200.

Case	Drag coefficient C_d			
	Johnson and Patel [25]	Gilmanov et al. [26]	White [27]	Present
Re = 100	1.112	1.153	1.18	1.128
Re = 200	0.79	–	0.81	0.8

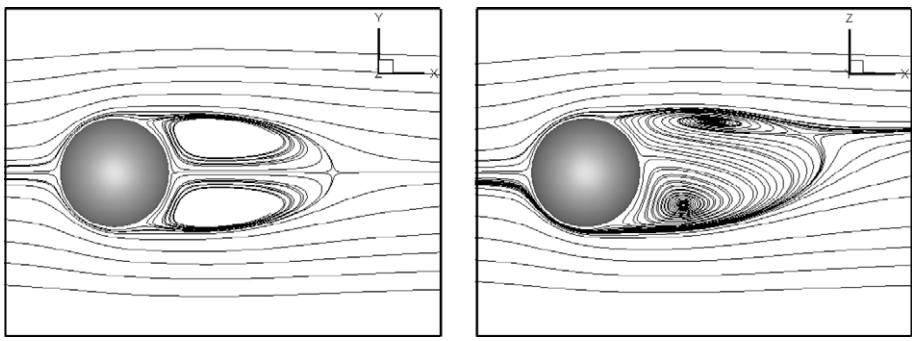

Fig. 9. Streamlines at Re = 250 for steady non-axisymmetric flow.

Here U_∞ is the free stream velocity and D is the sphere diameter. Traditionally, the sphere laminar flow could be classified into three different regimes: steady axisymmetric flow ($Re \leqslant 200$), steady non-axisymmetric flow ($210 \leqslant Re \leqslant 270$) and unsteady non-axisymmetric flow ($280 \leqslant Re < 375$).

In present simulation, 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 sphere is located at $(10D, 10D, 10D)$. A non-uniform mesh, which is fine and uniform around the sphere, is taken.

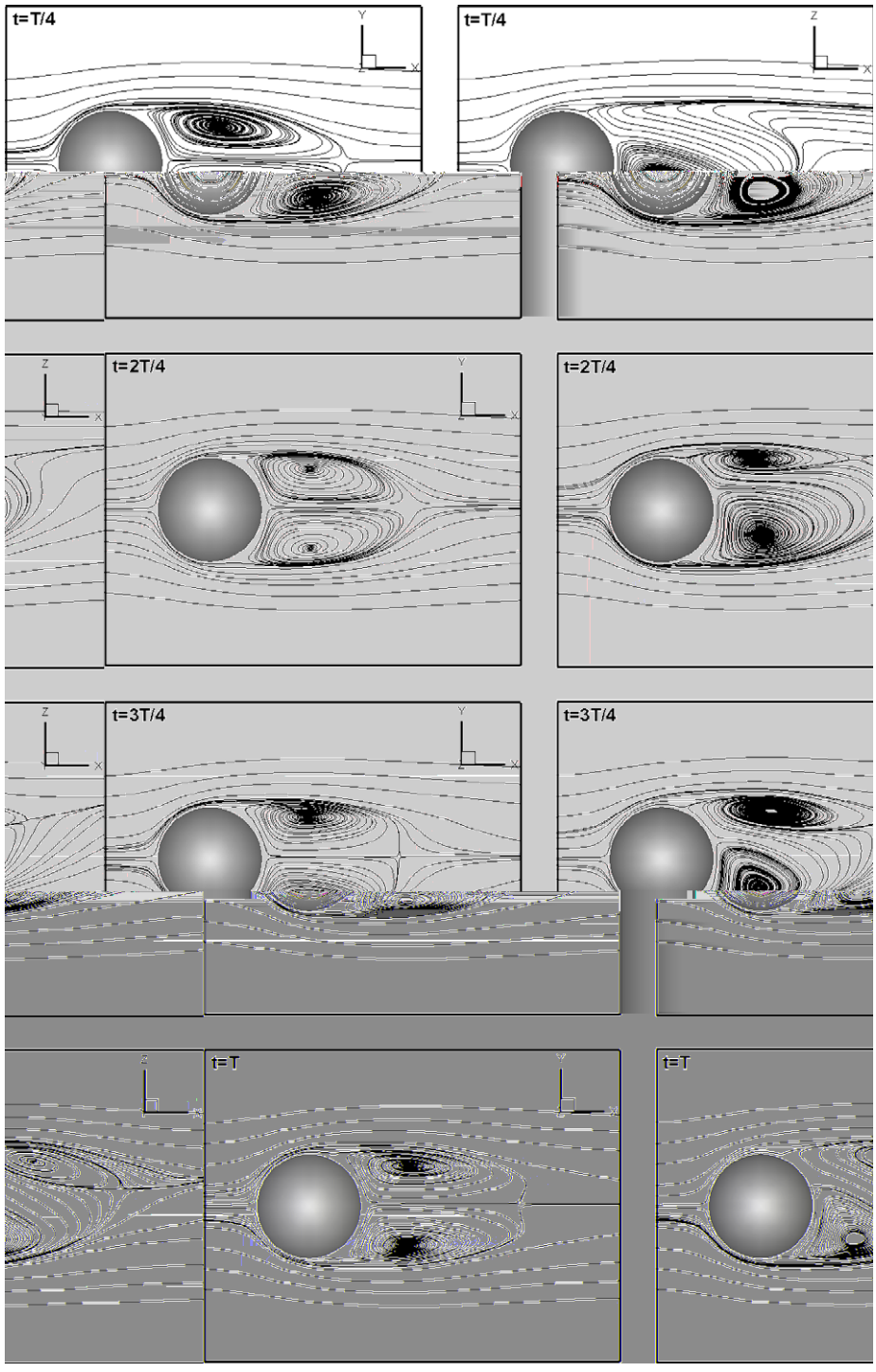

Fig. 10. Streamlines at Re = 300 for unsteady non-axisymmetric flow.

In current simulation, the fluid density is taken as $\rho = 1.0$ and the free stream velocity is $U_\infty = 0.1$. Initially, the free stream velocity is given in the whole computational domain. The equilibrium distribution functions are used to implement the outer boundary conditions.

To simulate the steady axisymmetric flow at Re = 50, 100, 150, and 200, the mesh size is chosen as $121 \times 101 \times 101$. The uniform mesh spacing around the sphere is taken as 0.025. The surface of sphere is discretized using triangular elements with 808 vertices.

Since the flow is axisymmetric, only the streamlines at the x–y plane of symmetry are plotted in Fig. 7. Since the boundary-layer separation happens for these Reynolds numbers, a recirculation region is appeared behind the sphere. This can be seen clearly in Fig. 7. It was found that the length of recirculation region (L_s) continuously increases with Reynolds number. Fig. 8 shows the variation of L_s with Re. For comparison, the results of Johnson and Patel [25] and Gilmanov et al. [26] are also included. Good agreement can be found in this figure. Table 1 compares the drag coefficients at Re = 100 and 200 with previous numerical [25,26] and experimental [27] data. Here, the drag coefficient C_d is defined as

$$C_d = \frac{F_D}{(1/2)\rho U_\infty^2 S} = \frac{8F_D}{\rho U_\infty^2 \pi D^2} \tag{49}$$

$$F_D = \int_\Gamma f_x^B \, d\Gamma \tag{50}$$

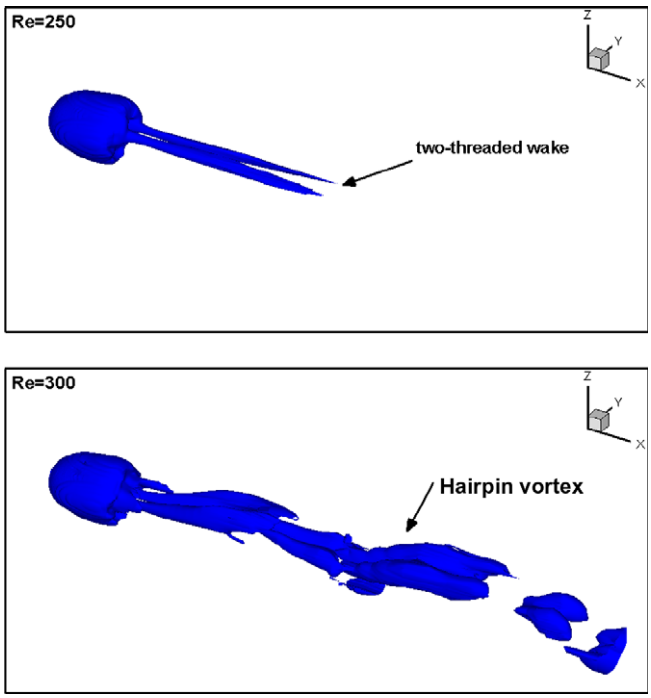

Fig. 11. 3D vortical structures of sphere for planar symmetric flows.

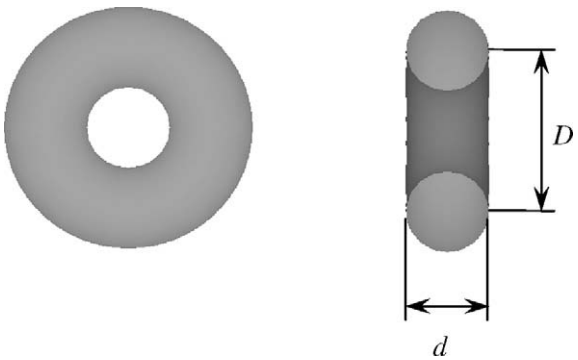

Fig. 12. Dimensions of the torus.

where f_x^B is the x-component of boundary force on the sphere. From the table we can see that the present numerical results compare very well with previous data. This validation shows that the improved IB-LBM can efficiently simulate three-dimensional flows with curved boundaries.

For steady and unsteady non-axisymmetric flow, the simulations at Re = 250 and 300 are carried out. The mesh size of $141 \times 121 \times 121$ is used. The uniform mesh spacing and the number of points on the sphere surface are the same as those for the axisymmetric flow simulations.

When the Reynolds number goes to 250, the flow over a sphere has undergone the regular asymmetric transition and becomes non-axisymmetric. This phenomenon is clearly displayed in Fig. 9, which shows the streamlines in the x–y and x–z planes of symmetry. It is apparent from figure that the symmetry is lost in the x–z plane, which implies that the flow is no longer axisymmetric. At the same time, the flow in the x–y plane is still symmetric. Hence, the flow at this situation can be considered as planar symmetric. These results are in good agreement with previous solutions [25,26]. When the Reynolds number is even higher and comes to 300, the flow undergoes a further transition, Hopf bifurcation, to unsteady state and the hairpin-shedding downstream from the sphere appears. Inferred from Fig. 10 which illustrates the streamlines in the x–y and x–z planes, the flow in the x–y plane maintains symmetric while the flow in the x–z plane becomes periodic. Therefore, this flow phenomenon can be recognized as unsteady planar symmetric. The Strouhal number in present simulation is 0.132 which is close to the values of 0.137 and 0.135 in [25] and [26], respectively. Using λ_2-method of Jeong and Hussain [28], 3D vortical structures behind the sphere are plotted in Fig. 11 for planar symmetric flows. In the case of Re = 250, a pair of vorticity tails stretches behind the sphere. This is the classic two-threaded wake which occurs after regular non-axisymmetric transition. In the case of Re = 300, the vorticity tails become unsteady, which is attributed to the non-axisymmetric Hopf transition. A hairpin vortex is formed in the wake. It can be observed from Fig. 11 that the vortical structures are symmetric to the x–z plane at this Reynolds number, although the axisymmetry is lost. However, if the Reynolds number continues to be increased and when $Re \geqslant 375$, the flow symmetry to the x–z plane would also be broken down. The flow in the wake becomes unsteady asymmetric. This case was not simulated in present study.

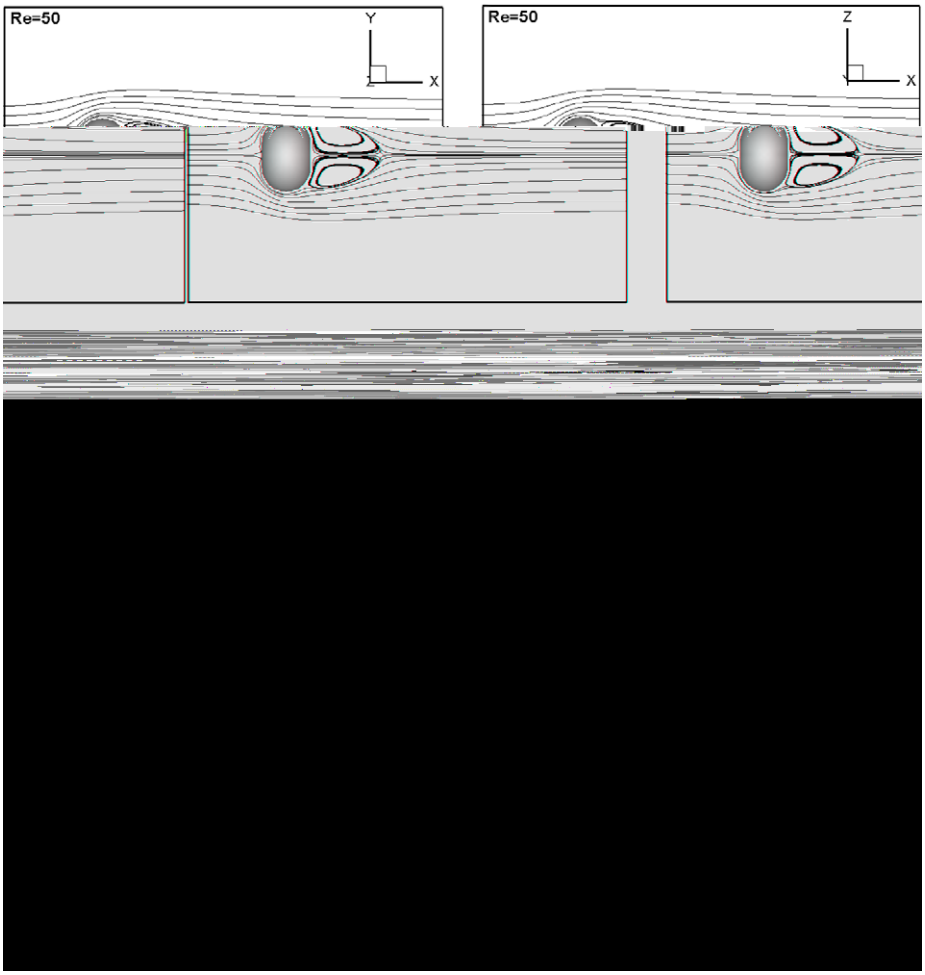

Fig. 13. Streamlines for flow over a torus with $Ar = 0.5$.

4.3. Flow over a small aspect ratio torus

To further validate the improved IB-LBM for 3D flows with complex geometry, the flow over a torus is simulated. Varying the aspect ratio (Ar) will give significantly different behaviors of the vortical structure behind the torus. Here, the aspect ratio is defined as $Ar = D/d$, where D is the mean torus diameter and d is the cross-section diameter, as shown in Fig. 12. If $Ar = 0$, the torus becomes a sphere, and when $Ar \to \infty$, the torus can be considered as a circular cylinder (just local to the cross-section of torus). For small Ar, the wake behind the torus is similar to that of sphere, while for sufficiently large Ar, the local wake from the cross-section of torus is very close to that of circular cylinder. As indicated by Sheard et al. [29,30], the critical Ar at which the flow over the torus switches from a circular cylinder type of shedding to a sphere type of hairpin wake occurs at $Ar \approx 3.9$.

In the present simulation, the torus is placed with its axis aligned with the flow direction. Two small aspect ratios are selected: $Ar = 0.5$ and 2. Note that the hole in the center of torus starts to appear when $Ar = 1$. According to Sheard et al. [29,30], the transition mode of torus with current two aspect ratios is similar to that of sphere. By increasing the Reynolds number, separation transition occurs firstly and regular non-axisymmetric transition occurs secondly, followed by Hopf transition. Here, the Reynolds number is based on the free stream velocity U_∞ and the cross-section diameter d. Like the sphere case, 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. The initial conditions and outer boundary conditions are implemented using the same way as used in the sphere case.

For $Ar = 0.5$, as shown by Sheard et al. [29], the critical Reynolds numbers corresponding to the three transitions are Re = 7, 123.4 and 152.4, respectively. Hence, three different Reynolds numbers are selected for the case of $Ar = 0.5$: Re = 50 (after separation transition), 150 (after regular non-axisymmetric transition) and 180 (after Hopf transition). To simulate the flow at these Reynolds numbers, the mesh size is chosen as $101 \times 101 \times 101$. The uniform mesh spacing around the torus is taken as 0.05. The surface of torus is discretized using triangular elements with 458 vertices.

Fig. 13 illustrates the streamlines at these Reynolds numbers. It is obvious that three distinct flow states can be found in Fig. 13: steady axisymmetric flow at Re = 50, steady planar symmetric flow at Re = 150 and unsteady planar symmetric flow at Re = 180. Actually, this flow behavior has a similarity to that of flow over a sphere as shown in Figs. 7, 9 and 10. For

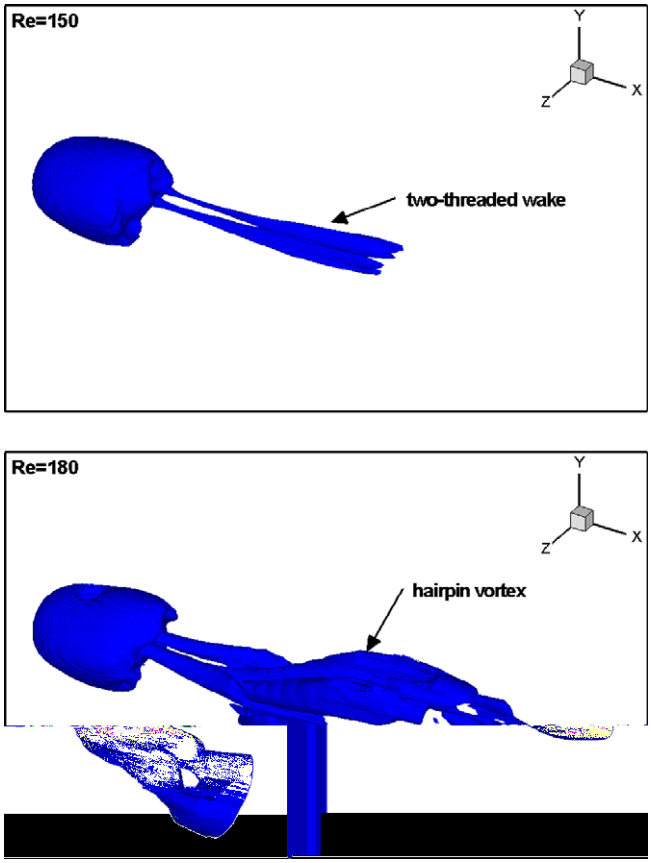

Fig. 14. Vortical structures behind torus without hole.

demonstration of 3D vortex, the vortical structures based on the method of Jeong and Hussain [28] are plotted in Fig. 14 for planar symmetric flows. The two-threaded wake and hairpin vortex behind the torus, for steady and unsteady planar symmetric flows, respectively, are observed. The phenomenon is the same as that of sphere, which can be found in Fig. 11.

In the case of $Ar = 2$, there is a hole in the center of torus. This means that the flow can go through the torus from its center portion, which is vastly different from flow over a sphere and torus with $Ar < 1$. Therefore, the wake behind the torus with $Ar = 2$ would exhibit some different flow behavior. Similar to the case of $Ar = 0.5$, three different Reynolds numbers are also selected for the case of $Ar = 2$: Re = 40 (after separation transition), 93 (after regular non-axisymmetric transition) and 120 (after Hopf transition). To simulate the flow at these Reynolds numbers, 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.

As a consequence, similar three distinct flow states also appear in Fig. 15 which displays the streamlines at three Reynolds numbers. However, because of the existence of hole in the center of torus for $Ar = 2$, the interesting phenomenon occurs. For steady axisymmetric flow in Fig. 15 (Re = 40), the recirculation region on the axis detaches from the rear part of torus, which is unlike the case of flow over a sphere and torus without hole (as shown in Figs. 7 and 13). Due to the presence of hole, the stagnation points of torus switch from the centerline to the cross-section surface. Hence, the new recirculation region appears behind the cross-section of torus. When the Reynolds number is increased, at the steady planar symmetric state (Re = 93), the recirculation region behind the cross-section of torus is enlarged. At the same time, the recirculation region on the axis is pushed further downstream and becomes smaller. Finally, for the unsteady planar symmetric flow (Re = 120), the recirculation region on the axis disappears. Fig. 16 plots the 3D vortical structure of planar symmetric flows. Again, λ_2-method of Jeong and Hussain [28] is employed. As seen clearly from Fig. 16, the familiar two-threaded wake and hairpin vortices, which can be observed from the case of flow over a sphere and torus with $Ar = 0.5$ (as shown in Figs. 11 and 14), do not appear.

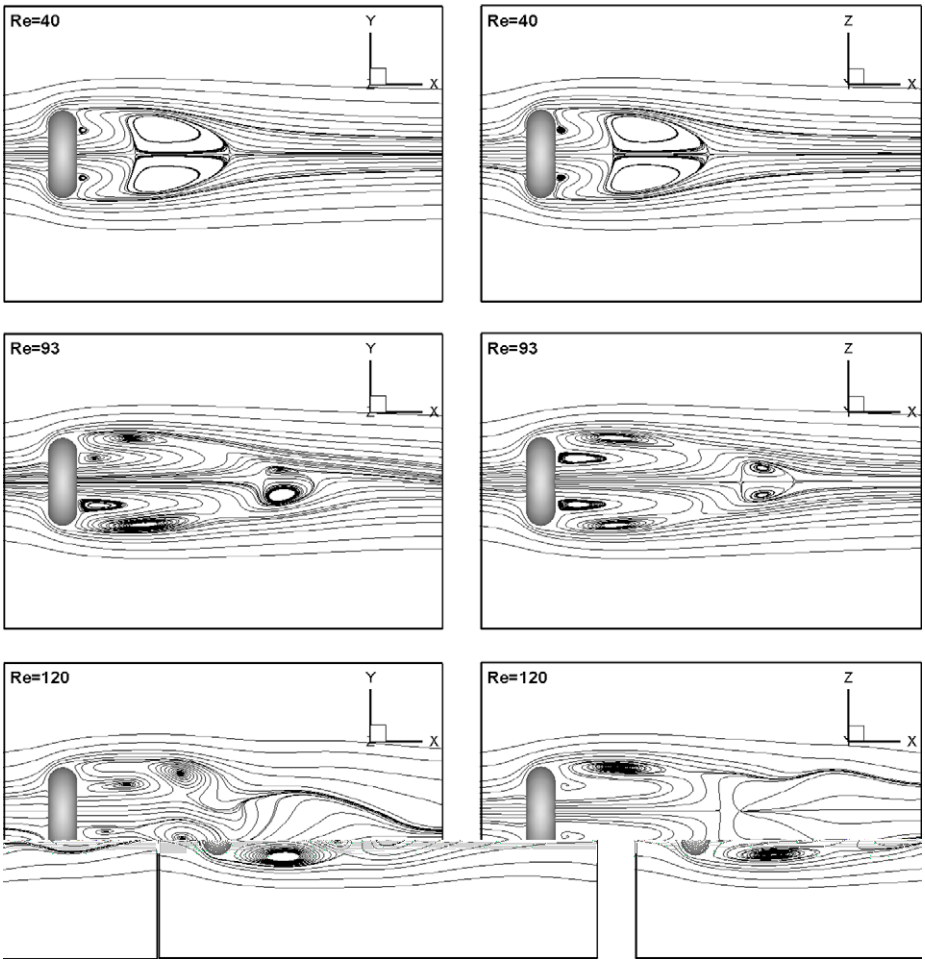

Fig. 15. Streamlines for flow over a torus with $Ar = 2$.

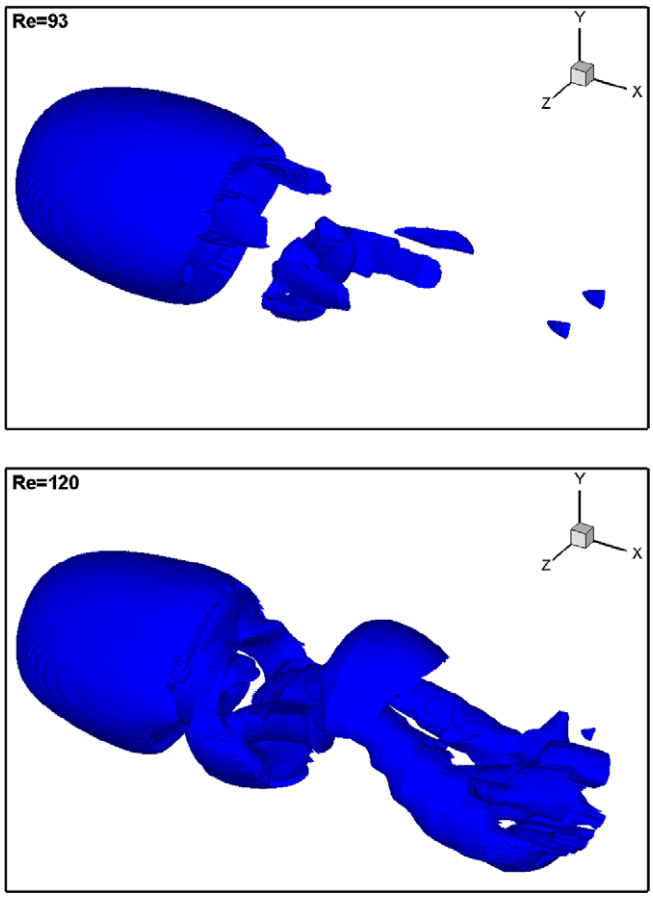

Fig. 16. Vortical structures behind torus with a hole.

Table 2
Drag coefficients of flow over torus.

Cases		Drag coefficient C_d	
		Present	Sheard et al. [31]
$Ar = 0.5$	Re = 50	1.338	–
	Re = 150	0.801	–
	Re = 180	0.745	–
$Ar = 2$	Re = 40	1.335	1.3
	Re = 93	1.013	1.0
	Re = 120	0.96	0.9

Different from the case of sphere, as suggested by Sheard et al. [31], the drag coefficient of torus can be defined as

$$C_d = \frac{2F_D}{\rho A_{frontal} U_\infty^2} \tag{51}$$

where $A_{frontal}$ is the projected frontal area of torus. If the dimensions are scaled by the diameter of torus cross-section d, $A_{frontal}$ can be simplified to a function of Ar

$$A_{frontal} = \begin{cases} \frac{4}{\pi(Ar^2+2Ar+1)} & 0 \leqslant Ar \leqslant 1 \\ \frac{1}{\pi Ar} & Ar > 1 \end{cases} \tag{52}$$

The obtained drag coefficients are listed in Table 2 for $Ar = 0.5$ at Re = 50, 150, 180 and for $Ar = 2$ at Re = 40, 93, 120, respectively. Sheard et al. [31] gave some results at different Ar and Re. For $Ar = 2$, their values of C_d are also included in Table 2 for comparison. It can be seen that the present numerical results compare favorably well with those of Sheard et al. [31].

5. Conclusions

The boundary condition-enforced immersed boundary-lattice Boltzmann method (IB-LBM) proposed in [14] can accurately simulate incompressible viscous flows around curved boundaries without flow penetration to the solid body. Usually, it is applied on the structured non-uniform Cartesian mesh. In this approach, the flow field is obtained by Taylor series expansion and least squares-based lattice Boltzmann method (TLLBM). TLLBM can be applied to any mesh point distribution but it needs considerable memory or time to store or compute the weighting coefficients. This drawback greatly limits the application of TLLBM and IB-LBM for simulation of three-dimensional flows. To overcome this drawback, an efficient LBM solver, which is specifically developed for IB-LBM on the non-uniform Cartesian mesh, is proposed in this paper. The new LBM solver is actually based on the one-dimensional interpolation along the straight mesh lines. Its weighting coefficients can be simply calculated by algebraic formulations without solving an equation system, and the number of weighting coefficients is much less than that in TLLBM. Numerical example of simulating the three-dimensional cubic cavity flow showed that the proposed approach takes about 8% of computational time of TLLBM. The high efficiency of the approach is demonstrated.

The computational efficiency of boundary condition-enforced IB-LBM [14] is significantly improved when TLLBM is replaced by the proposed LBM solver on the non-uniform Cartesian mesh. Numerical examples for simulating flows around a sphere and torus demonstrated that the improved IB-LBM can accurately and effectively simulate the general three-dimensional viscous flows with curved boundaries.

In the present work, only the flow around stationary boundaries is considered. For this case, the matrix **A** in Eq. (18) and its inverse do not change with time. So, we can calculate them once and store for the following computations. It is indicated that when the flow around a moving boundary is considered, the matrix **A** and its inverse will change with time. Therefore, we have to calculate them at every time step.

Acknowledgment

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

References

[1] H. Johansen, P. Colella, A Cartesian grid embedded boundary method for Poisson's equation on irregular domains, J. Comput. Phys. 147 (1998) 60–85.
[2] H.S. Udaykumar, R. Mittal, P. Rampunggoon, A. Khanna, A sharp interface Cartesian grid method for simulating flows with complex moving boundaries, J. Comput. Phys. 174 (2001) 345–380.
[3] E.A. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, J. Comput. Phys. 161 (2000) 35–60.
[4] A. Gilmanov, F. Sotiropoulos, A hybrid Cartesian/immersed boundary method for simulating flows with 3D, geometrically complex, moving bodies, J. Comput. Phys. 207 (2005) 457–492.
[5] L. Ge, F. Sotiropoulos, A numerical method for solving the 3D unsteady incompressible Navier–Stokes equations in curvilinear domains with complex immersed boundaries, J. Comput. Phys. 225 (2007) 1782–1809.
[6] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220–252.
[7] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (2005) 239–261.
[8] S. Chen, G.D. Doolen, Lattice Boltzmann method for fluid flows, Annu. Rev. Fluid Mech. 30 (1998) 329–364.
[9] S. Succi, The Lattice Boltzmann Equation, For Fluid Dynamics and Beyond, Oxford University Press, 2001.
[10] Z. Feng, E. Michaelides, The immersed boundary-lattice Boltzmann method for solving fluid–particles interaction problems, J. Comput. Phys. 195 (2004) 602–628.
[11] Z. Feng, E. Michaelides, Proteus: a direct forcing method in the simulations of particulate flows, J. Comput. Phys. 202 (2005) 20–51.
[12] X.D. Niu, C. Shu, Y.T. Chew, Y. Peng, A momentum exchange-based immersed boundary-lattice Boltzmann method for simulating incompressible viscous flows, Phys. Lett. A 354 (2006) 173–182.
[13] Y. Peng, C. Shu, Y.T. Chew, X.D. Niu, X.Y. Lu, Application of multi-block approach in the immersed boundary-lattice Boltzmann method for viscous fluid flows, J. Comput. Phys. 218 (2006) 460–478.
[14] J. Wu, C. Shu, Implicit velocity correction-based immersed boundary-lattice Boltzmann method and its applications, J. Comput. Phys. 228 (2009) 1963–1979.
[15] X. He, G.D. Doolen, Lattice Boltzmann method on a curvilinear coordinate system: vortex shedding behind a circular cylinder, Phys. Rev. E 56 (1997) 434–440.
[16] R. Mei, W. Shyy, On the finite difference-based lattice Boltzmann method in curvilinear coordinates, J. Comput. Phys. 143 (1998) 426–448.
[17] G. Peng, H. Xi, C. Duncan, S.H. Chou, Finite volume scheme for the lattice Boltzmann method on unstructured meshes, Phys. Rev. E 59 (1999) 4675–4682.
[18] D. Yu, R. Mei, W. Shyy, A multi-block lattice Boltzmann method for viscous fluid flows, Int. J. Numer. Meth. Fluid 39 (2002) 99–120.
[19] C. Shu, X.D. Niu, Y.T. Chew, Taylor-series expansion and least-squares-based lattice Boltzmann method: two-dimensional formulation and its applications, Phys. Rev. E 65 (2002) 036708.
[20] X.D. Niu, Y.T. Chew, C. Shu, Simulation of flows around an impulsively started circular cylinder by Taylor series expansion- and least squares-based lattice Boltzmann method, J. Comput. Phys. 188 (2003) 176–193.
[21] Y.H. Qian, D. d'Humieres, P. Lallemand, Lattice BGK models for Navier–Stokes equation, Europhys. Lett. 17 (1992) 479–484.
[22] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479–517.
[23] V. Babu, S.A. Korpela, Numerical solution of the incompressible three-dimensional Navier–Stokes equations, Comput. Fluid 23 (1994) 675–691.
[24] C. Shu, X.D. Niu, Y.T. Chew, Taylor series expansion and least squares-based lattice Boltzmann method: three-dimensional formulation and its applications, Int. J. Mod. Phys. C 14 (2003) 925–944.
[25] T.A. Johnson, V.C. Patel, Flow past a sphere up to a Reynolds number of 300, J. Fluid Mech. 378 (1999) 19–70.
[26] A. Gilmanov, F. Sotiropoulos, E. Balaras, A general reconstruction algorithm for simulating flows with complex 3D immersed boundaries on Cartesian grids, J. Comput. Phys. 191 (2003) 660–669.
[27] F.M. White, Viscous Fluid Flow, McGraw-Hill, New York, 1974.

[28] J. Jeong, J. Hussain, On the identification of a vortex, J. Fluid Mech. 285 (1995) 69–94.
[29] G.J. Sheard, M.C. Thompson, K. hourigan, From spheres to circular cylinders: the stability and flow structures of bluff ring wakes, J. Fluid Mech. 492 (2003) 147–180.
[30] G.J. Sheard, M.C. Thompson, K. Hourigan, From spheres to circular cylinders: non-axisymmetric transitions in the flow past rings, J. Fluid Mech. 506 (2004) 45–78.
[31] G.J. Sheard, K. Hourigan, M.C. Thompson, Computations of the drag coefficients for low-Reynolds-number flow past rings, J. Fluid Mech. 526 (2005) 257–275.