Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Physics 226 (2007) 1607–1622

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

A novel immersed boundary velocity correction–lattice Boltzmann method and its application to simulate flow past a circular cylinder

C. Shu *, N. Liu, Y.T. Chew

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

Received 6 July 2006; received in revised form 31 May 2007; accepted 5 June 2007
Available online 14 June 2007

Abstract

A novel immersed boundary velocity correction–lattice Boltzmann method is presented and validated in this work by its application to simulate the two-dimensional flow over a circular cylinder. The present approach is inspired from the conventional immersed boundary method (IBM). In the conventional IBM, the effect of rigid body on the surrounding flow is modeled through a forcing term, which is in turn used to correct the surrounding velocity field. It was found that this process is actually an iterative procedure, trying to satisfy the non-slip boundary condition at the solid wall. In this work, a new concept of immersed boundary velocity correction approach is proposed, which directly corrects the velocity to enforce the physical boundary condition. The main advantage of the new method is that it is simple in concept and easy for implementation, and the convergence of numerical computation is faster and more stable than the conventional IBM. One challenging issue of conventional IBM is that some streamlines may pass through the solid body since there is no mechanism to enforce the non-slip condition at the boundary. As shown in the present numerical results, this unphysical phenomenon is avoided in our new method since the non-slip condition is enforced. The present results for the steady and unsteady flows compare very well with available data in the literature.
© 2007 Elsevier Inc. All rights reserved.

Keywords: Incompressible flow; Immersed boundary method; Velocity correction; Lattice Boltzmann method; Flow around cylinder; Cartesian mesh

1. Introduction

The ability to handle complex geometries accurately and efficiently has been the primary issue in computational fluid dynamics. Conventional approaches such as finite-difference and finite-element methods are generally used to accommodate complex geometries with tedious grid generation. In contrast, the recently developed immersed boundary method (IBM) can handle complex geometry with the use of Cartesian mesh.

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

0021-9991/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2007.06.002

The concept of IBM was introduced by Peskin [1] in the 1970s in order to model the blood flow in the heart. The IBM comes from the concept that the deformation or moving of the boundary will yield a force that tends to restore the boundary to its original shape or position. The restoring forces on the boundary are in turn distributed into the surrounding nodes and the flow field with a body force is solved over the whole fluid domain including both the inside and outside of immersed body. This is the so-called virtual boundary force IBM.

Based on the work of Peskin [1], numerous research works have been done to modify or refine the IBM [2–10]. Among the remarkable works, Goldstein et al. [2] proposed a model named virtual boundary method which permits simulations with complex geometries. Lai and Peskin [3] proposed a second-order accurate immersed boundary method with adoption of a well-chosen Dirac delta function. Linnick and Fasel [4] proposed a high-order modified immersed interface method for the two-dimensional, unsteady, incompressible Navier–Stokes equations in the stream function–vorticity formulation, which employs an explicit fourth-order Runge–Kutta scheme for time integration and the fourth-order compact finite-difference schemes for approximation of spatial derivatives. Lima E Silva et al. [5] proposed a version named physical virtual model, which is based on the conservation laws, and simulated an internal channel flow and the flow around a circular cylinder. Due to the common feature of using Cartesian mesh in the lattice Boltzmann method (LBM) and IBM, some researchers tried to combine these two methods into an efficient one. The first such attempt was made by Feng and Michaelides [6,7]. In their work, the restoration force due to deformation is computed by the penalty method [6] or the direct-forcing scheme [7]. The penalty method introduces a user-defined spring parameter which may have a significant effect on the computational efficiency and accuracy. The direct-forcing method was introduced firstly by Fadlun et al. [11] in order to overcome the drawbacks of the virtual boundary force method. Kim et al. [8] later proposed an explicit variant of the direct-forcing method which allows maintaining the simple matrix structure of a standard finite-difference method. The direct-forcing immersed boundary method and its improved versions [12] are very suitable for finite difference applications to simulate flows in complex domains. However, the direct-forcing scheme requires solving the N–S equations by the finite-difference method, which may spoil the merits of LBM.

The key issue in the virtual boundary force IBM is to compute the restoring force. One of the popular ways to do it is from the difference between the boundary (Lagrangian point) velocity and the local fluid (Eulerian point) velocity. The velocity difference at the same point produces a displacement. According to the Hooke's law, the restoration force due to the deformation is modeled by a linear spring relation, where the spring constant in the calculation should make the particle surface stiff enough for the displacement to be small but not large enough to affect the convergence of the computations.

From the process to compute the restoring force, it is obvious that IBM is an iterative procedure to satisfy both the governing equation and the boundary condition. Suppose that at an intersection point, the boundary velocity and the fluid velocity are the same. Then the non-slip boundary condition is satisfied, and there is no displacement. According to the Hooke's law, the restoring force is zero. Therefore, we can say that if the converged solution is achieved, the local restoring force should be zero in theory. On the other hand, we notice that in the iterative process of gradually decreasing the restoring force, the non-slip condition is not directly enforced. Instead, it is a part of the solution. Due to numerical errors, at the converged state, the non-slip condition is only approximately satisfied. As a result, some streamlines may pass through the solid body. This is not true in physics. To remove this drawback, Kim et al. [8] introduced the mass source or sink into the computation. Using this way, the non-slip condition can be well kept, but the complexity is introduced into the computation.

Having understood that IBM is an iterative process to satisfy the boundary condition, in this work, we present a new approach to directly enforce the non-slip boundary condition. Our approach is very simple and straightforward. In our approach, in the vicinity of the boundary point, the velocity correction is made to enforce the boundary condition directly, and the basic velocity field is obtained by LBM on the Cartesian mesh. For simplicity, we term the new approach as immersed boundary velocity correction method (IBVCM), in which, the ad hoc coefficient and the force calculation are not required. To validate the proposed method, numerical simulation of two-dimensional flows past a circular cylinder is performed. Both steady and unsteady cases are considered. The obtained numerical results agree very well with available data in the literature. Since the non-slip boundary condition can be accurately satisfied, the obtained streamlines do not pass through the solid body.

2. Immersed boundary velocity correction–lattice Boltzmann method

The immersed boundary velocity correction method is developed from the conventional immersed boundary method (IBM). In the following, we will briefly describe the conventional IBM first, and then introduce the immersed boundary velocity correction method, which directly enforces the non-slip boundary condition.

2.1. Conventional immersed boundary method (IBM)

In the conventional IBM, the solid body is immersed in the flow field. The effect of body on the flow field is through the body force distributed from the restoring force on the solid boundary. For a two-dimensional problem, the governing equation for the fluid motion can be written as

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \tag{1}$$

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial x}+\upsilon\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)+f_x \tag{2}$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=-\frac{1}{\rho}\frac{\partial p}{\partial y}+\upsilon\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)+f_y \tag{3}$$

Here u, v are the velocity components in the x and y directions, p is the pressure, ρ is the density and v is the viscosity. f_x and f_y are, respectively, the components of body force f in the x and y directions. The body force f is distributed from the restoring force $\boldsymbol{F}$ at the boundary, which is usually given by the following form:

$$f(x,t)=\oint \boldsymbol{F}(s,t)\delta(\boldsymbol{x}-\boldsymbol{X}(s,t))\,\mathrm{d}s \tag{4}$$

where $\boldsymbol{X}(s,t)$, $0\leqslant s\leqslant L_{\mathrm{b}}$ represents the boundary coordinate, $\boldsymbol{x}$ is the coordinate of Eulerian point, δ is the delta function [3]. Using Hooke's law, the restoring force at a boundary point can be determined by

$$\boldsymbol{F}(s,t)=-k\Delta\xi=-k(\boldsymbol{V}_{\text{fluid}}\Delta t-\boldsymbol{V}_{\text{wall}}\Delta t) \tag{5}$$

where $\boldsymbol{V}_{\text{fluid}}$ is the fluid velocity at the boundary point interpolated from the surrounding fluid points, $\boldsymbol{V}_{\text{wall}}$ is the boundary velocity of the body, k is the spring coefficient.

2.2. Immersed boundary velocity correction method (IBVCM)

Using the fractional step technique, the solution of Eqs. (1)–(3) can be obtained by the following two steps:

Step 1 (*prediction step*): Solve the normal Navier–Stokes equations (setting $f_x=0$, $f_y=0$ in Eqs. (1)–(3)) and obtain the predicted velocity components u^* and v^*.
Step 2 (*correction step*): Correct the velocity field by solving

$$\frac{\partial u}{\partial t}=f_x \tag{6}$$

$$\frac{\partial v}{\partial t}=f_y \tag{7}$$

There are many numerical schemes to solve Eqs. (6) and (7). When the Euler explicit scheme is applied, we have

$$u^{n+1}=u^*+\Delta t\cdot f_x \tag{8}$$

$$v^{n+1}=v^*+\Delta t\cdot f_y \tag{9}$$

where Δt is the time step size, u^* and v^* are the solution of step 1 obtained from u^n and v^n at the previous time level. Note that in the above process, u^n, v^n, u^{n+1}, v^{n+1} satisfy Eqs. (1)–(3) but u^* and v^* only satisfy Navier–Stokes equations with zero body force.

In the conventional IBM, f_x and f_y are computed from Eqs. (4) and (5). When the corrected velocity field is obtained from Eqs. (8) and (9), there is no guarantee that the velocity $\boldsymbol{V}_{\text{fluid}}$ interpolated from the corrected velocity field satisfies the non-slip boundary condition, that is, $\boldsymbol{V}_{\text{fluid}} = \boldsymbol{V}_{\text{wall}}$. This is because the restoring force and the resultant body force f_x and f_y are all pre-determined. To overcome this drawback, the immersed boundary velocity correction method (IBVCM) is introduced. The basic idea of IBVCM is that the body forces f_x and f_y are not pre-determined. Instead, f_x and f_y should satisfy a constraint. That is, the velocity $\boldsymbol{V}_{\text{fluid}}$ interpolated from the corrected velocity field given by Eqs. (8) and (9) satisfies the non-slip boundary condition, $\boldsymbol{V}_{\text{fluid}} = \boldsymbol{V}_{\text{wall}}$. For simplicity, we can define the velocity correction u' and v' as

$$u' = \Delta t \cdot f_x \tag{10}$$

$$v' = \Delta t \cdot f_y \tag{11}$$

As a result, Eqs. (8) and (9) can be written as

$$u^{n+1} = u^* + u' \tag{12}$$

$$v^{n+1} = v^* + v' \tag{13}$$

Now, the problem is simplified to determine the velocity correction u' and v' so that the condition $\boldsymbol{V}_{\text{fluid}} = \boldsymbol{V}_{\text{wall}}$ can be satisfied. Obviously, in the above process, the computation of restoring force and the resultant body force is avoided.

In general, the computation of $\boldsymbol{V}_{\text{fluid}}$ from the corrected velocity field involves the two-dimensional interpolation, which may be complicated. On the other hand, it can be observed from Eqs. (8) and (9) that the x-component of the body force f_x only corrects the x-component velocity u while the y-component of the body force f_y only corrects the y-component velocity v. This information is very important. As shown in Fig. 1, the Cartesian mesh has two intersection points M_x and M_y with the immersed boundary. As the numerical computation only involves the functional value at the mesh point, the intersection point M_x only affects the u-velocity correction along the horizontal mesh line while the intersection point M_y only affects the v-velocity correction along the vertical mesh line. Like the conventional IBM, we limit the effect of the intersection point to a local region in the flow domain. As shown in Fig. 1, it is supposed that the intersection point M_x only affects the u-velocity correction at two mesh points A and B, and the intersection point M_y only affects the v-velocity correction at two mesh points B and C. Here, A and C are in the flow domain while B is inside the body. Clearly, at point B, both the u-velocity correction and v-velocity correction are needed, while at point A, only the u-velocity correction is needed. Similarly, at point C, only the v-velocity correction is needed. As shown below, the velocity correction can be determined easily.

The velocity components at points A, B and C can be denoted, respectively, as (u_A, v_A), (u_B, v_B) and (u_C, v_C). The interpolated velocity from the flow domain is denoted as (u_{ix}, v_{ix}) for the intersection point M_x (by

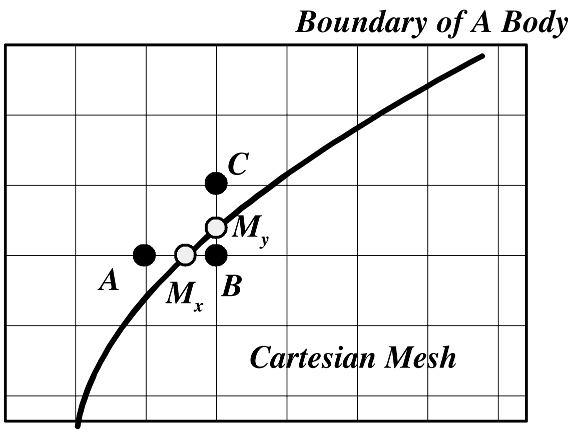

Fig. 1. Cartesian mesh lines, immersed boundary and their intersection points.

interpolation in the x direction) and (u_{iy}, v_{iy}) for the intersection point M_y (by interpolation in the y direction). The wall velocity is represented by (u_{Mx}, v_{Mx}) for the intersection point M_x and (u_{My}, v_{My}) for the intersection point M_y. The distance between points A and B (mesh spacing in the x direction) is denoted as $\overline{AB} = \Delta x$, and the distance between points A and M_x is $\overline{AM_x} = \varDelta_1$.

As shown in Fig. 2, the u-velocity between points A and B has a linear relationship. Thus, it is easy to get the interpolated velocity at the intersection point M_x as

$$u_{ix} = u_A + \frac{\varDelta_1}{\Delta x}(u_B - u_A) \tag{14}$$

Note that the computation of v-velocity at M_x is not necessary as there is no need to correct the v-velocity at points A and B. The interpolated velocity u_{ix} given by Eq. (14) may not be the same as the given boundary condition u_{Mx}. So, we have to correct u_{ix} to u_{Mx}. As shown in Fig. 2, this can be easily done by moving the x axis to the new position x'. As a consequence, all the u-velocity between A and B would be corrected by $u_{Mx} - u_{ix}$. In other words, the velocity correction between the points A and B is

$$u' = u_{Mx} - u_{ix} = u_{Mx} - u_A - \frac{\varDelta_1}{\Delta x}(u_B - u_A) \tag{15}$$

Obviously, when the u-velocity at the points A and B is corrected by Eq. (15), the interpolated velocity at the intersection point M_x satisfies the boundary condition $u = u_{Mx}$.

In a similar manner, only the v-velocity correction is needed between the points B and C, which can be written as

$$v' = v_{My} - v_B - \frac{\varDelta_2}{\Delta y}(v_C - v_B) \tag{16}$$

where v_{My} is the v-velocity component of the wall velocity at the intersection point M_y, Δy is the distance between the points B and C, $\Delta y = \overline{BC}$ (mesh spacing in the y direction), and $\varDelta_2$ is the distance between the points B and M_y, $\varDelta_2 = \overline{BM_y}$.

With Eqs. (15) and (16), the computation of IBVCM is very simple. The basic time marching procedure can be summarized below:

(1) Solve Navier–Stokes equations with zero body force to get the intermediate velocity field u^* and v^*.

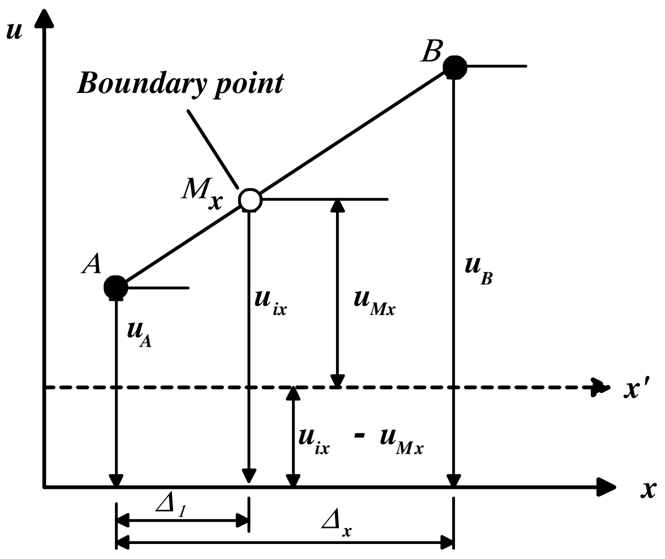

Fig. 2. Linear velocity distribution between two mesh points.

(2) Apply Eqs. (15) and (16) to correct the velocity at those mesh points next to the boundary positions which are the intersection points between the Cartesian mesh lines and the immersed boundary.
(3) Repeat steps (1) and (2) until the convergence threshold is satisfied.

Fig. 1 shows the normal situation of intersection points between the Cartesian mesh lines and the immersed boundary, in which the u-velocity at B is corrected by the information at the intersection point M_x and the v-velocity at B is corrected by the information at the intersection point M_y. Some special cases of intersection points will be discussed in the following sub-section.

2.3. Some special cases for velocity correction

It is noticed that for some cases, the immersed boundary may have more than one intersection points with the Cartesian mesh lines within a mesh spacing. An example is shown in Fig. 3, where the immersed boundary has three intersection points with the horizontal mesh lines within the mesh spacing, Δx. For this case, more velocity correction is needed to ensure the non-slip boundary condition being satisfied. Fig. 3 shows that the immersed boundary has one intersection point M_y with the vertical mesh line, and four intersection points M_1, M_2, M_3, M_4 with the horizontal mesh lines. Among them, the points M_2, M_3 and M_4 are within the mesh spacing Δx. The points B and E are inside the solid body. As the distances between the point B and the intersection points M_y and M_2 are within the mesh spacing Δx and Δy, the velocity correction at B can be done using the same way as described in the above section. That is, the u-velocity correction between A and B is made by Eq. (15), while the v-velocity correction between B and C is made by Eq. (16). As for the point E, its distance to the intersection point M_3 on the horizontal mesh line is less than the mesh spacing Δx. So, the u-velocity correction between the points D and E can be made by using a similar equation to Eq. (15), which will be shown below. However, the distance from E to the intersection point M_y on the vertical mesh line is larger than the mesh spacing Δy. This may result in a larger interpolation error, and lead to inconsistent v-velocity correction at the point C as its correction value can be obtained in the interval between B and C and can also be computed in the interval between E and C. On the other hand, it was found from the numerical experiment that the v-velocity correction at E is necessary. Otherwise, it may lead some streamlines to pass through the solid body because the non-slip boundary condition is not accurately satisfied. To enforce the non-slip

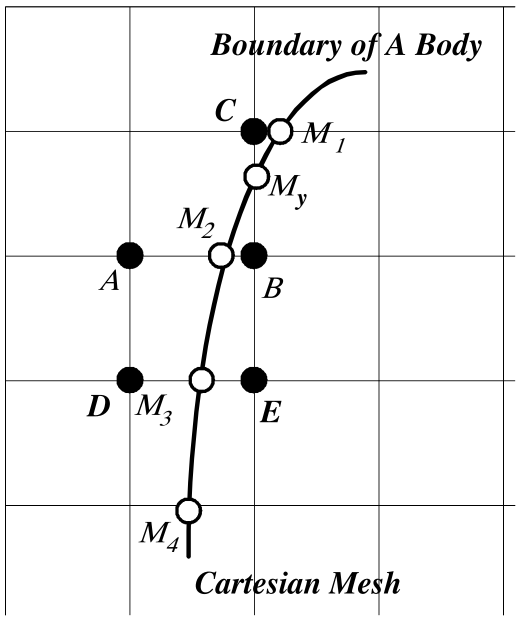

Fig. 3. A special case with more than one intersection points within Δx.

boundary condition at wall, and in the meantime, to keep consistent velocity correction at the mesh points, both the u-velocity correction and v-velocity correction are made between D and E using the following equations:

$$u' = u_{M_3} - u_D - \frac{\Delta_1}{\Delta x}(u_E - u_D) \tag{17}$$

$$v' = v_{M_3} - v_D - \frac{\Delta_1}{\Delta x}(v_E - v_D) \tag{18}$$

where (u_{M_3}, v_{M_3}) is the wall velocity at the intersection point M_3, (u_D, v_D) and (u_E, v_E) are, respectively, the intermediate velocity at points D and E, Δ_1 is the distance between D and M_3.

In practice, we may encounter another case. That is, the immersed boundary may have more than one intersection points with vertical mesh lines within the mesh spacing Δy. This can be seen clearly in Fig. 4, where the intersection points M_2, M_3 and M_4 are within Δy, and the points B and D are inside the solid body. The velocity correction for the point B can be done normally by using Eqs. (15) and (16). The u-velocity correction and v-velocity correction between D and E can be made by the following equations:

$$u' = u_{M_3} - u_D - \frac{\Delta_1}{\Delta y}(u_E - u_D) \tag{19}$$

$$v' = v_{M_3} - v_D - \frac{\Delta_1}{\Delta y}(v_E - v_D) \tag{20}$$

where (u_{M_3}, v_{M_3}), (u_D, v_D) and (u_E, v_E) and Δ_1 have the same meaning as the previous case.

2.4. Implementation by lattice Boltzmann method

In the IBVCM, the intermediate velocity components u^* and v^* in Eqs. (12) and (13) are the solution of incompressible Navier–Stokes (N–S) equations with zero value of body force. When the primitive variable form of N–S equations is used, the solution of Poisson equation is needed. Usually, the convergence rate for the solution of Poisson equation is not very fast. Sometimes, it is even difficult to get converged solution. Recently, the lattice Boltzmann method (LBM) [13–22] was developed into an alternative promising approach for simulation of incompressible viscous flows. The major advantage of LBM is its simplicity and easy implementation. In this work, LBM is used as a tool to give the predicted (intermediate) velocity components u^* and v^* on the Cartesian mesh. The lattice Boltzmann equation can be written as

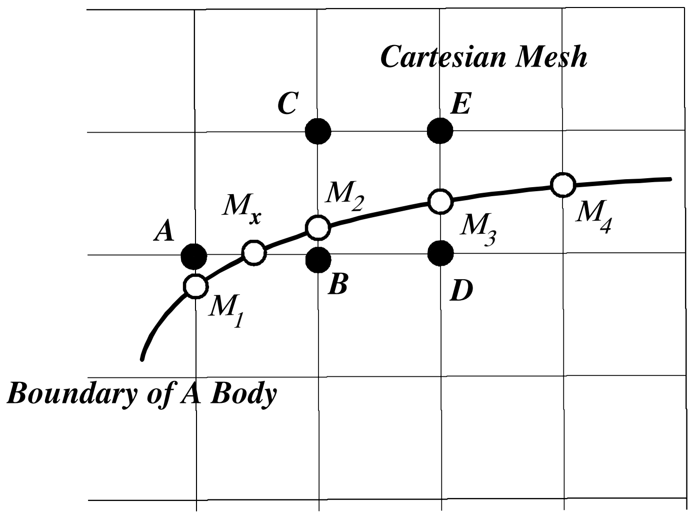

Fig. 4. A special case with more than one intersection points within Δy.

$$f_\alpha(\boldsymbol{x}+\boldsymbol{e}_\alpha\delta t, t+\delta t) - f_\alpha(\boldsymbol{x},t) = -\frac{1}{\tau}\left(f_\alpha(\boldsymbol{x},t) - f_\alpha^{\mathrm{eq}}(\boldsymbol{x},t)\right) \tag{21}$$

where τ is the single relaxation time; f_α is the distribution function; f_α^{eq} is its corresponding equilibrium state; δt is the time step; $\boldsymbol{e}_\alpha$ is the particle velocity. In the LBM computation, the D2Q9 lattice velocity model is used, which is defined as

$$\boldsymbol{e}_\alpha = \begin{cases} 0 & \alpha = 0 \\ (\cos[(\alpha-1)\pi/2], \sin[(\alpha-1)\pi/2]) & \alpha = 1,2,3,4 \\ \sqrt{2}(\cos[(\alpha-5)\pi/2+\pi/4], \sin[(\alpha-5)\pi/2+\pi/4]) & \alpha = 5,6,7,8 \end{cases} \tag{22}$$

Accordingly, the equilibrium distribution function is given by

$$f_\alpha^{\mathrm{eq}} = w_\alpha \rho\left[1 + 3\boldsymbol{e}_\alpha\cdot\boldsymbol{V} + \frac{9(\boldsymbol{e}_\alpha\cdot\boldsymbol{V})^2}{2} - \frac{3\boldsymbol{V}^2}{2}\right] \tag{23}$$

where $w_0 = 4/9$, $w_\alpha = 1/9$ for $\alpha = 1,2,3,4$, $w_\alpha = 1/36$ for $\alpha = 5,6,7,8$. Note that f_α^{eq} depends on the macroscopic density and velocity. The corrected velocity field in the previous time level should be used in f_α^{eq}. Clearly, the effect of corrected velocity field on the distribution function f_α is through f_α^{eq}.

From conservation laws of mass and momentum, the macroscopic density ρ and fluid velocity $\boldsymbol{V}$ are calculated in terms of the density distribution functions as

$$\rho = \sum_{\alpha=0}^{8} f_\alpha, \quad \boldsymbol{V} = \frac{1}{\rho}\sum_{\alpha=0}^{8} f_\alpha \boldsymbol{e}_\alpha \tag{24}$$

The pressure p can be calculated from the equation of state, and for the D2Q9 model, it is given by

$$p = \frac{\rho}{3} \tag{25}$$

As shown above, the distribution function f_α is affected by the corrected velocity field through f_α^{eq}. As a result, according to Eqs. (24) and (25), the obtained pressure is also affected by the corrected velocity field.

The kinematic viscosity υ can be linked to the relaxation time τ in Eq. (21) through the Chapman–Enskog expansion in such a way that the macroscopic variables obtained by LBM satisfy the N–S equations. For the D2Q9 model, the relationship between υ and τ is given by

$$\upsilon = \frac{1}{3}\left(\tau - \frac{1}{2}\right)\delta t \tag{26}$$

It should be indicated that the LBM is used in IBVCM as a tool only to provide the predicted velocity field. Since no body force is involved in this step, the use of LBM is exactly the same as its application for normal flow problems. In other words, we do not need to modify the lattice Boltzmann equation. This is different from the combination of conventional IBM with LBM, which does need to modify the lattice Boltzmann equation in order to consider the effect of the body force. In this sense, we can say that the implementation of IBVCM is simpler than the conventional IBM.

3. Results and discussions

In order to examine the accuracy and efficiency of the proposed immersed boundary velocity correction method (IBVCM), numerical simulations of the viscous flow past a circular cylinder are carried out. Both the steady and unsteady cases are considered. This problem has been studied extensively and a number of numerical and experimental results exist in literature [9,23–33]. The problem is very attractive because the flow behavior depends on the Reynolds number, $Re = u_\infty D/\upsilon$, and is not easy to simulate accurately using Cartesian grids. Here, u_∞ is the free stream velocity, v is the kinematic viscosity of fluid. To well capture the high velocity gradients around the cylinder surface, the non-uniform mesh is used, and the Taylor series expansion and least square-based lattice Boltzmann method (TLLBM) [13] is applied to obtain the predicted velocity field. The computational domain is set by $L \times W = 42D \times 42D$ (square domain). The circular cylinder is located at the center of computational domain. The free steam velocity u_∞ is set to be 0.1, and the free stream density ρ_∞ is taken as 1. For all the simulations, the whole mesh size is 561×561, and in the region including the immersed body, $-0.52 \leqslant x \leqslant 0.52$, $-0.52 \leqslant y \leqslant 0.52$, a uniform fine mesh of 361×361 with the minimum mesh spacing of 0.00289 is used. In other regions, the mesh spacing gradually increases towards the outer boundary. For this problem, the drag and lift coefficients are good parameters to validate the accuracy of numerical results obtained by various numerical methods. In this study, the drag force F_D and lift force F_L on the immersed body are calculated by using the control volume method. Integration of momentum equations over a rectangular control volume Ω gives

$$F_\mathrm{D} = -\left\{\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_\Omega u_1\,\mathrm{d}\Omega\right) + \int_S \left[u_1\vec{u}\cdot\vec{n} + pn_1 - \mu\left(\frac{\partial u_1}{\partial x_i} + \frac{\partial u_i}{\partial x_1}\right)\right]\mathrm{d}s\right\} \tag{27}$$

$$F_\mathrm{L} = -\left\{\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_\Omega u_2\,\mathrm{d}\Omega\right) + \int_S \left[u_2\vec{u}\cdot\vec{n} + pn_2 - \mu\left(\frac{\partial u_2}{\partial x_i} + \frac{\partial u_i}{\partial x_2}\right)\right]\mathrm{d}s\right\} \tag{28}$$

where S is the control surface, $\vec{n}$ is the normal vector to the boundary of control surface, $\vec{n} = (n_1, n_2)$, the subscripts 1 and 2 denote the x-direction and y-direction, respectively. Note that the use of Eqs. (27) and (28) is exactly the same as their applications in conventional fluid mechanics. That is, the volume Ω is the flow domain, which does not include the interior of the cylinder, and the surface S represents the outer surface of the control volume, which does not include the cylinder surface. The effect of the cylinder surface is reflected by the forces F_D and F_L. The drag and lift coefficients are defined by

$$C_\mathrm{D} = \frac{F_\mathrm{D}}{0.5\rho u_\infty^2 D}, \quad C_\mathrm{L} = \frac{F_\mathrm{L}}{0.5\rho u_\infty^2 D} \tag{29}$$

The pressure coefficient on the cylinder surface is defined as:

$$\mathrm{Cp} = \frac{p_\mathrm{w} - p_\infty}{0.5\rho u_\infty^2} \tag{30}$$

Here, p_w is the pressure on the boundary and p_∞ is the free stream pressure. Like the conventional IBM, the obtained pressure profile on the cylinder surface shows some oscillation. To remove this difficulty, we adopted the way suggested by Kim et al. [8] to compute the wall pressure. That is, the pressure is computed on a closed surface which is 3/4 mesh spacing away from the cylinder surface.

In the TLLBM computation to obtain the predicted velocity field, only the far field boundary is involved. In this work, the equilibrium density distribution function with the given free stream velocity and density is used to implement the boundary condition at the far field boundary. For the unsteady flow simulation where the Karman vortex street occurs, the velocity and density at the far field boundary of downstream can be determined by extrapolation from the flow field. At the cylinder surface, the non-slip (zero velocity) boundary condition is imposed, which is used in IBVCM to correct the velocity field at the mesh points near the cylinder boundary (see Eqs. (15) and (16)).

In the conventional IBM, the boundary effect is through the interpolation between the boundary (Lagrangian) points and the mesh (Eulerian) points, while in the IBVCM, the boundary effect is through the intersection points M_x and M_y at the boundary which are determined by the mesh lines and the boundary curve. Note that in the IBM, the boundary points and the mesh points are independent. In IBVCM, M_x and M_y do not depend on the distribution of boundary points. They are dependent of the local mesh lines and the boundary curve. For the general case, the boundary curve may not be expressed by a simple function. Instead, for the two-dimensional case, it can be represented by a series of line segments. For this case, M_x and M_y can be determined by intersection points of mesh lines with all line segments. In this work, the boundary curve (cylinder surface) is represented by 800 line segments (800 boundary points) for both IBM and IBVCM computations.

3.1. Steady flow past a circular cylinder

For the numerical simulation, the initial fluid density is set as $\rho = 1.0$ and the initial velocity is taken as the free stream velocity u_∞. Extensive simulations were performed for the steady case.

It was found that as compared to the conventional IBM, the present method has a faster convergence rate to reach the steady state solution. This can be clearly observed in Figs. 5 and 6. From numerical experiments, it was found that the spring coefficient in IBM has very little effect on the convergence rate. In this study, its value of 32 is used for all the numerical examples. Fig. 5 shows the time evolution of drag coefficients at $Re = 40$, while Fig. 6 displays the early stage of convergence history at the same Reynolds number of 40. Obviously, the present results and the conventional IBM results will eventually converge to the same value, but the present method has a faster convergence rate. As shown in Fig. 6, at the non-dimensional time $T = 1$, the drag coefficient obtained by the present method converges to the steady state resolution. However, at this time, the drag coefficient obtained by the conventional IBM still shows the fluctuation feature. It approaches the steady state solution at $T = 3$. The faster convergence of the present method may be due to the fact that the non-slip boundary condition is accurately satisfied in IBVCM. In the conventional IBM, the non-slip boundary condition is not enforced. One just expects that when the converged solution is obtained, the non-slip condition can be approximately satisfied. It may take more iteration steps for satisfying the non-slip condition. It should

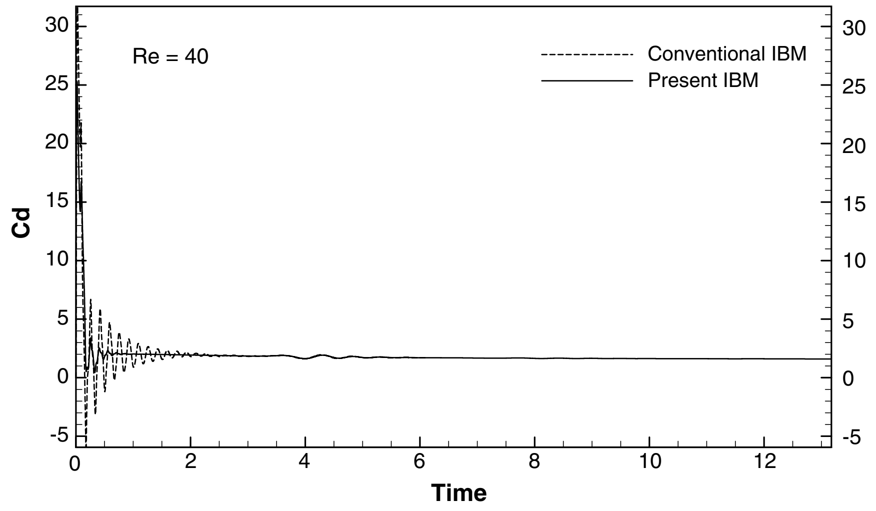

Fig. 5. Whole convergence history of drag coefficient at $Re = 40$.

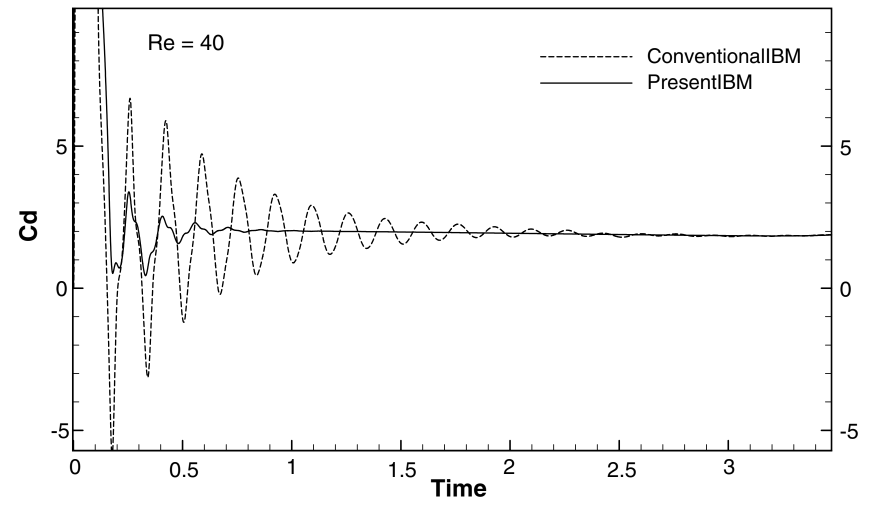

Fig. 6. Early stage of convergence history of drag coefficient at $Re = 40$.

be indicated that the fast convergence feature of a numerical method is very important for the simulation of unsteady flows when the explicit time marching technique is adopted.

Apart from the faster convergence feature, the CPU time required by the IBVCM at each time step is slightly less than that required by the conventional IBM. This is because IBVCM does not involve δ-function interpolation from the Eulerian points to Lagrangian points or from Lagrangian points to Eulerian points. However, since most of the CPU time is taken by LBM computation which is the same for two methods, the reduction of CPU time is very little. For example, for the case of $Re = 20$, at each time step, the CPU time needed by IBVCM and conventional IBM is, respectively, 2.74 and 2.79 s.

The accuracy of IBVCM results is also very good. Fig. 7 shows the distribution of drag coefficient versus the Reynolds number in logarithmic scale. Obviously, the present results compare very well with the experimental data [28] and other numerical results [5,32]. Table 1 compares the non-dimensional length of recirculating eddy obtained by the present method and other research work at $Re = 20$ and 40. Here, the non-dimensional length of recirculating eddy is defined as $2L/D$, and the length of the recirculating region, L, is measured from the rearmost point of the cylinder to the end of the wake. From Table 1, we can see that our numerical results of the recirculating length are in good agreement with previous ones [19,23–26,29]. The streamlines of steady state resolution obtained by the present method at $Re = 20$ and 40 are depicted in Fig. 8. Clearly, a pair of symmetric eddy is developed behind the cylinder. As shown in Fig. 8, unlike the conventional IBM results, the present results do not reveal any streamline to pass through the cylinder. This demonstrates the high accuracy of present method in satisfying the non-slip boundary condition. Indeed, when the converged solution is obtained, the velocity on the cylinder surface is zero or very close to zero. This can be con-

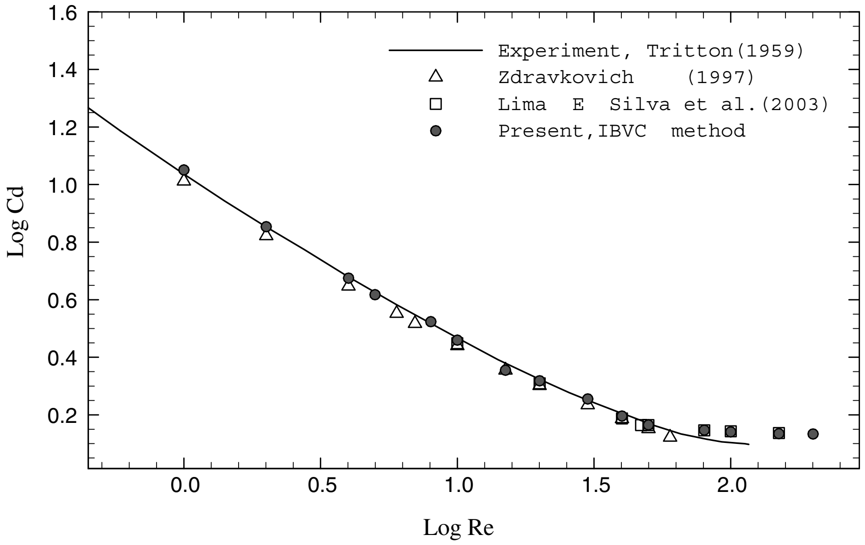

Fig. 7. Comparison of drag coefficients.

Table 1
Comparison of recirculating length ($2L/D$) with previous studies

Authors	$Re = 20$	$Re = 40$
He and Doolen [19]	1.842	4.49
Nieuwstadt and Keller [23]	1.786	4.357
Coutanceau and Bouard [24]	1.86	4.26
Dennis and Chang [25]	1.88	4.69
Fornberg [26]	1.82	4.48
Tseng and Ferziger [29]	–	4.42
Present	1.80	4.40

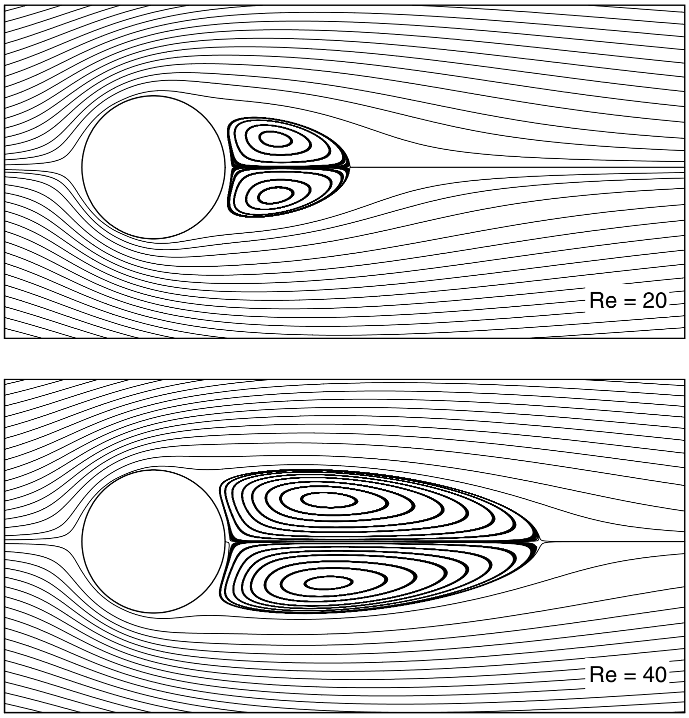

Fig. 8. Streamlines at final steady state for Reynolds numbers of 20 and 40.

firmed in Fig. 9, which shows the contour of $|\vec{u}| = 0$ (magnitude of velocity) at $Re = 40$ in the whole computational domain. Although the cylinder surface is not given, it is well revealed by the contour of $|\vec{u}| = 0$ in Fig. 9.

The pressure profile for $Re = 40$ is shown in Fig. 10. Also included in Fig. 10 are the numerical and experimental results of He and Doolen [19] and Park et al. [34]. The horizontal axis θ is the orientation angle, which is changed along the boundary from 0 (leading stagnation point) to 180 (trailing stagnation point). It can be seen from Fig. 10 that the present results basically agree well with those in the literature.

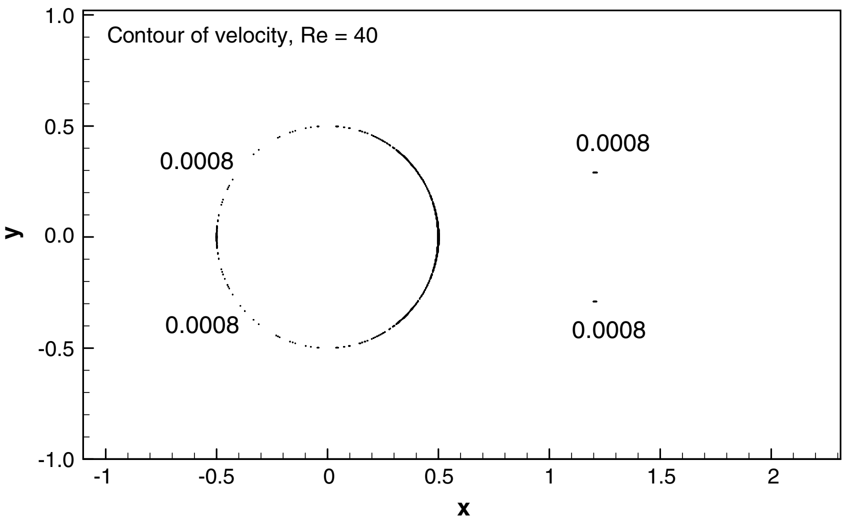

Fig. 9. The contours of $|\vec{u}| = 0$ at $Re = 40$.

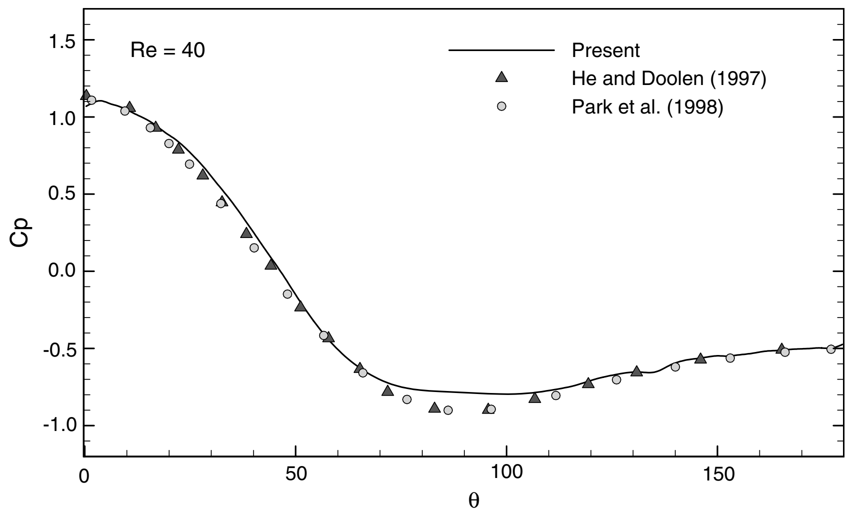

Fig. 10. The pressure profile at $Re = 40$.

Table 2
Comparison of average drag coefficient and lift coefficient for $Re = 100$

	C_D (average)	C_L
Lai and Peskin [3]	1.4473	±0.3299
Kim et al. [8]	1.33	±0.32
Tseng and Ferziger [29]	1.42	±0.29
Liu et al. [35]	1.35	±0.339
Present	1.3833	±0.35

3.2. *Unsteady flow past a circular cylinder*

To further validate the IBVCM, the unsteady flow around the cylinder at moderate Reynolds numbers was simulated. In this case, the initial perturbation in the flow field would trigger the alternating separation of the

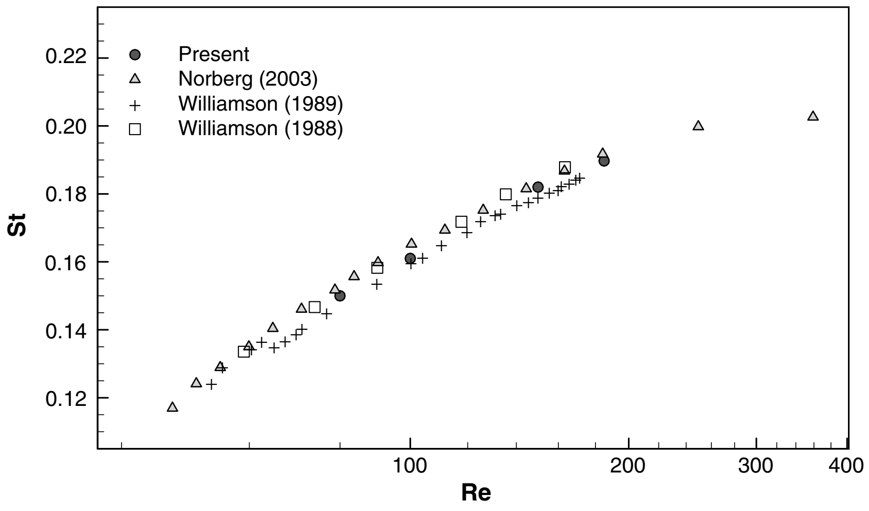

Fig. 11. Distribution of Strouhal number versus Reynolds number (logarithmic scale is used in the horizontal axis).

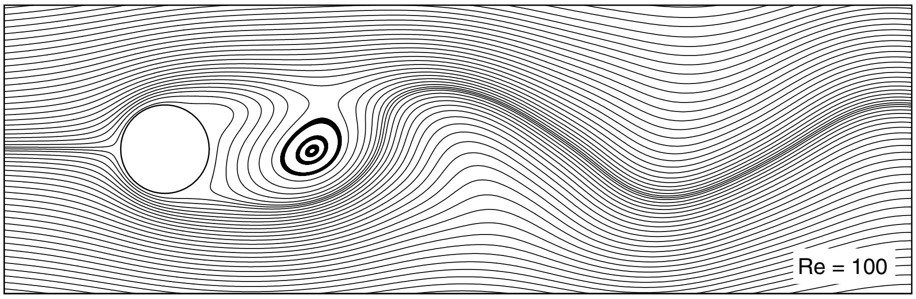

Fig. 12. Near wake structures of instantaneous streamlines for $Re = 100$.

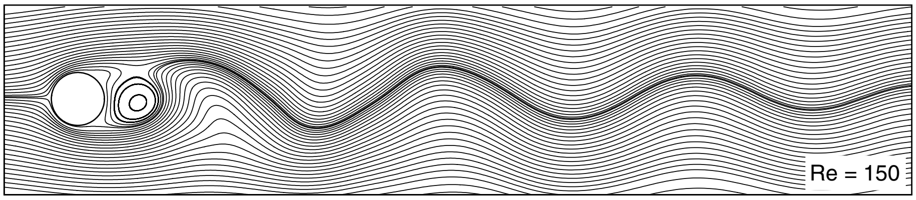

Fig. 13. Near wake structures of instantaneous streamlines at $Re = 150$.

vortices which form the well-known Karman vortex street. Eventually, the flow becomes periodically unsteady. It was found that the present method can well predict the vortex street behind the cylinder. The logarithm of average drag coefficient versus the logarithm of Reynolds number is shown in Fig. 7. It can be seen from this figure that the present results agree very well with the experimental data of Tritton [28] and the numerical results of Lima E Silva et al. [5] and Zdravkovich [32]. Table 2 compares the averaged value of drag and lift coefficients at $Re = 100$ between our results and those from other researchers [3,8,29,35]. Good agreement is shown in this table. The Strouhal number, which is defined as

$$St = fD/u_\infty \tag{31}$$

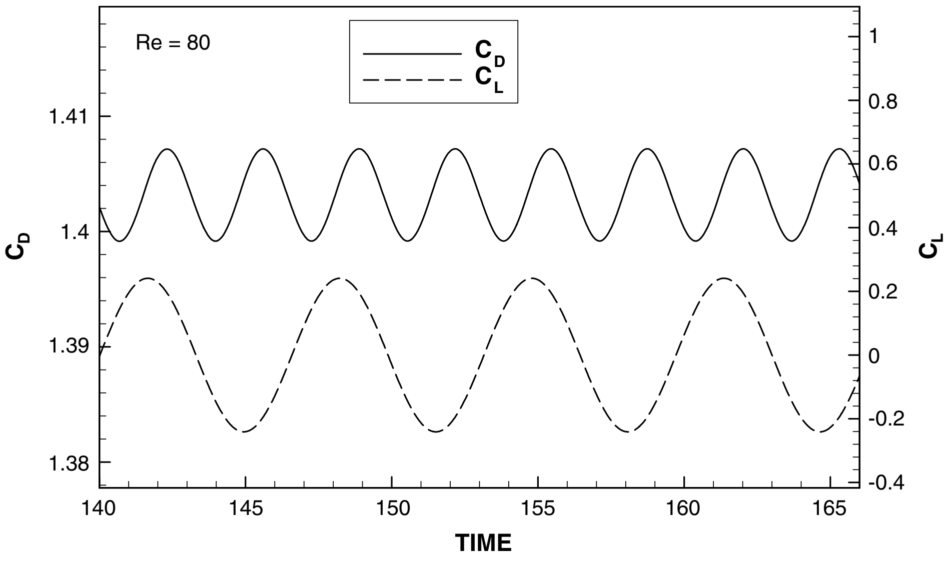

Fig. 14. Lift and drag coefficients at $Re = 80$.

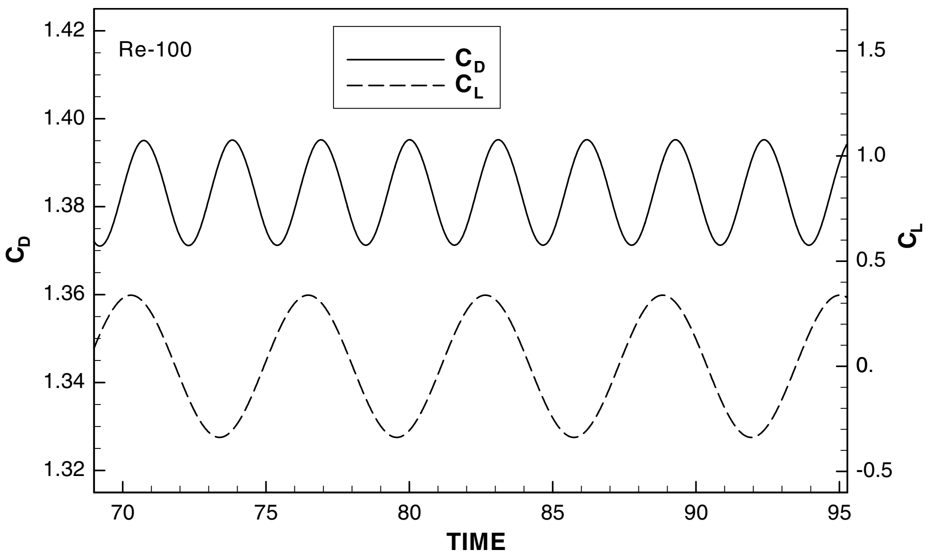

Fig. 15. Lift and drag coefficients at $Re = 100$.

is a good parameter to measure the oscillation of fluid flow in the wake region. Here f is the shedding frequency of vortex. The Strouhal number computed by the present method is displayed in Fig. 11. Also included in this figure are the results of Williamson [30,31] and Norberg [27]. Clearly, the present results have a very good agreement with the published data. Note that the logarithmic scale is used in the horizontal-axis (Reynolds number) of Fig. 11, following the work of Norberg [27]. Figs. 12 and 13 show the instantaneous streamlines of Karman vortex street at $Re = 100$ and 150. It can be seen clearly from these figures that even for an unsteady flow, the present results do not show any streamline passing through the cylinder. This further demonstrates the high accuracy of the present method in implementing the non-slip boundary condition.

Figs. 14 and 15 present the time evolution of drag and lift coefficients at $Re = 80$ and 100. The periodicity of flow pattern is clearly revealed. It can be seen from these figures that the period of drag coefficient is different from that of lift coefficient. The period of lift coefficient is about two times the period of drag coefficient. The amplitude of lift coefficient increases notably as the Reynolds number increases. In contrast, the amplitude of drag coefficient has a little variation.

4. Conclusions

A new immersed boundary velocity correction method (IBVCM) is developed in this paper. IBVCM adopts the idea of the conventional immersed boundary method (IBM). That is, the effect of the immersed boundary on the flow field is through the body force in the momentum equations. From the concept of the fractional step technique, adding a body force in the momentum equations is equivalent to make a correction in the velocity field. In the IBVCM, there is no calculation of the restoring force, and the velocity correction is made in such a way that the velocity at the boundary interpolated from the corrected velocity field satisfies the non-slip boundary condition. In the present computation, the predicted velocity field is obtained by using the lattice Boltzmann method (LBM).

The accuracy and efficiency of the proposed method are examined by its application to simulate two-dimensional steady and unsteady flows past a circular cylinder. Numerical results showed that the present method has a faster convergence rate than the conventional IBM. The obtained results are in good agreement with available data in the literature. In particular, the streamlines obtained by IBVCM do not penetrate the solid body. It seems that the present method has a great potential in simulation of steady and unsteady flows.

References

[1] C.S. Peskin, Numerical analysis of blood flow in the heart, J. Comput. Phys. 25 (1977) 220.
[2] D. Goldstein, R. Hadler, L. Sirovich, Modeling a no-slip flow boundary with an external force field, J. Comput. Phys. 105 (1993) 354.
[3] M. Lai, C.S. Peskin, An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, J. Comput. Phys. 160 (2000) 705.
[4] M.N. Linnick, H.F. Fasel, A high-order immersed interface method for simulating unsteady incompressible flows on irregular domains, J. Comput. Phys. 204 (2005) 157.
[5] A.L.F. Lima E Silva, A. Silveira-Neto, J.J.R. Damasceno, Numerical simulation of two-dimensional flows over a circular cylinder using the immersed boundary method, J. Comput. Phys. 189 (2003) 351.
[6] Z.G. Feng, E.E. Michaelides, The immersed boundary-lattice Boltzmann method for solving fluid-particles interaction problems, J. Comput. Phys. 195 (2004) 602.
[7] Z.G. Feng, E.E. Michaelides, Proteus: a direct forcing method in the simulations of particulate flows, J. Comput. Phys. 202 (2005) 20.
[8] J. Kim, D. Kim, H. Choi, An immersed-boundary finite-volume method for simulations of flow in complex geometries, J. Comput. Phys. 171 (2001) 132.
[9] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (2005) 239.
[10] C.S. Peskin, The immersed boundary method, Acta Numer. 11 (2002) 479.
[11] E. 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.
[12] M. Uhlmann, An immersed boundary method with direct forcing for the simulation of particulate flows, J. Comput. Phys. 209 (2005) 448.
[13] C. Shu, X.D. Niu, Y.T. Chew, Taylor series expansion- and least square-based lattice Boltzmann method: two-dimensional formulation and its applications, Phys. Rev. E 65 (2002) 036708.
[14] S. Chen, G.D. Doolen, Lattice Boltzmann method for fluid flows, Annu. Rev. Fluid Mech. 30 (1998) 329.
[15] H. Chen, Volumetric formulation of the lattice Boltzmann method for fluid dynamics: basic concept, Phys. Rev. E 58 (1998) 3955.
[16] X. Shan, H. Chen, Lattice Boltzmann model for simulating flows with multiple phases and components, Phys. Rev. E 47 (3) (1993) 1815.
[17] S. Succi, The Lattice Boltzmann Equation for Fluid Dynamics and Beyond, Oxford University Press, 2001, p. 62 (Chapter 4).
[18] S. Hou, Q. Zou, G. Doolen, A. Cogley, Simulation of cavity flow by the lattice Boltzmann method, J. Comput. Phys. 118 (1995) 329.
[19] X. He, G. Doolen, Lattice Boltzmann method on curvilinear coordinates system: flow around a circular cylinder, J. Comput. Phys. 134 (1997) 306.
[20] T. Inamuro, T. Ogata, S. Tajima, N. Konishi, A lattice Boltzmann method for incompressible two-phase flows with large density differences, J. Comput. Phys. 198 (2004) 628.
[21] P. Lallemand, L.-S. Luo, Theory of the lattice Boltzmann method: dispersion, dissipation, isotropy, Galilean invariance, and stability, Phys. Rev. E 61 (2000) 6546.
[22] T. Lee, C.-L. Lin, A stable discretization of the lattice Boltzmann equation for simulation of incompressible two-phase flows at high density ratio, J. Comput. Phys. 206 (2005) 16.
[23] F. Nieuwstadt, H.B. Keller, Viscous flow past circular cylinders, Comput. Fluid 1 (1973) 59.
[24] M. Coutanceau, R. Bouard, Experimental determination of the main features of the viscous flow in the wake of circular cylinder in uniform translation. Part I. Steady flow, J. Fluid Mech. 79 (1977) 231.
[25] S.C.R. Dennis, G.Z. Chang, Numerical solutions for steady flow past a circular cylinder at Reynolds number up to 100, J. Fluid Mech. 42 (1970) 471.
[26] B. Fornberg, A numerical study of steady viscous flow past a circular cylinder, J. Fluid Mech. 98 (1980) 819.
[27] C. Norberg, Fluctuating lift on a circular cylinder: review and new measurements, J. Fluid Struct. 17 (2003) 57.
[28] D.J. Tritton, Experiments on the flow past a circular cylinder at low Reynolds number, J. Fluid Mech. 6 (1959) 547.
[29] Y.-H. Tseng, J.H. Ferziger, A ghost-cell immersed boundary method for flow in complex geometry, J. Comput. Phys. 192 (2003) 593.
[30] C.H.K. Williamson, Defining a universal and continuous Strouhal–Reynolds number relationship for the laminar vortex shedding of a circular cylinder, Phys. Fluid 31 (1988) 2742.
[31] C.H.K. Williamson, Oblique and parallel models of vortex shedding in the wake of a circular cylinder at low Reynolds numbers, J. Fluid Mech. 206 (1989) 579.
[32] M.M. Zdravkovich, Flow Around Circular Cylinders (v1), Oxford University Press, 1997, p. 245.
[33] T. Ye, R. Mittal, H.S. Udaykumar, W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex boundaries, J. Comput. Phys. 156 (1999) 209.
[34] J. Park, K. Kwon, H. Choi, Numerical solutions of flow past a circular cylinder at Reynolds number up to 160, KSME Int. J. 12 (1998) 1200.
[35] C. Liu, X. Zheng, C.H. Sung, Preconditioned multigrid methods for unsteady incompressible flows, J. Comput. Phys. 139 (1998) 35.