Computers & Fluids 38 (2009) 370–381

Contents lists available at ScienceDirect

Computers & Fluids

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

Robust treatment of no-slip boundary condition and velocity updating for the lattice-Boltzmann simulation of particulate flows

Zhi-Gang Feng [a*], Efstathios E. Michaelides [b]

[a] Mechanical and Energy Engineering, University of North Texas, Denton, TX 76203, USA
[b] Department of Mechanical Engineering, University of Texas, San Antonio, TX 78259, USA

ARTICLE INFO

Article history:
Received 2 February 2007
Received in revised form 28 January 2008
Accepted 23 April 2008
Available online 15 May 2008

ABSTRACT

In the past decade, the lattice-Boltzmann method (LBM) has emerged as a very useful tool in studies for the direct-numerical simulation of particulate flows. The accuracy and robustness of the LBM have been demonstrated by many researchers; however, there are several numerical problems that have not been completely resolved. One of these is the treatment of the no-slip boundary condition on the particle–fluid interface and another is the updating scheme for the particle velocity. The most common used treatment for the solid boundaries largely employs the so-called "bounce-back" method (BBM). [Ladd AJC. Numerical simulations of particulate suspensions via a discretized Boltzmann equation Part I. Theoretical foundation. J Fluid Mech (1994);271:285; Ladd AJC. Numerical simulations of particulate suspensions via a discretized Boltzmann equation Part II. Numerical results. J Fluid Mech (1994);271:311.] This often causes distortions and fluctuations of the particle shape from one time step to another. The immersed boundary method (IBM), which assigns and follows a series of points in the solid region, may be used to ensure the uniformity of particle shapes throughout the computations. To ensure that the IBM points move with the solid particles, a force density function is applied to these points. The simplest way to calculate the force density function is to use a direct-forcing scheme. In this paper, we conduct a complete study on issues related to this scheme and examine the following parameters: the generation of the forcing points; the choice of the number of forcing points and sensitivity of this choice to simulation results; and, the advantages and disadvantages associated with the IBM over the BBM. It was also observed that the commonly used velocity updating schemes cause instabilities when the densities of the fluid and the particles are close. In this paper, we present a simple and very effective velocity updating scheme that does not only facilitate the numerical solutions when the particle to fluid density ratios are close to one, but also works well for particle that are lighter than the fluid.

1. Introduction

The beginnings of LBM may be traced in the 1980s with the work of Frish et al. [3]. After the contributions of Ladd [1,2] on the treatment of solid boundaries and those of several others, the LBM has become a powerful and frequently used computational method for particulate flows, largely because of its simplicity and computational efficiency. The LBM uses a fixed regular grid, avoiding the time-consuming grid adapting; the velocity and fluid properties are computed using local values, making it suitable for massive parallel computing. However, there are two main issues that have not been satisfactorily resolved. One is the treatment of the boundary condition on the surface of a solid particle and the other, the instability of updating the particle motion for light particles.

The "bounce-back" method (BBM), first proposed by Ladd [1,2] classifies the computational nodes as fluid nodes and solid nodes. The fluid nodes are embedded in the fluid, while the solid nodes appear in the interior of the solid particles. To facilitate the computations, the fluid particle distribution functions for both the fluid and solid nodes follow the LBM governing equations. The BBM assumes that when fluid moves from the fluid nodes toward the solid nodes, the fluid mater will be reflected or "bounce back" to the fluid nodes. The reflections occur at the mid-point of the links between the fluid and solid nodes. The particle surface is represented by the so-called "boundary nodes," which are essentially the set of mid-points of these links between two fixed grids, with one of the grids being within the fluid domain and the other within the solid domain.

This arrangement causes the computational boundary of a particle to be defined by a stepwise boundary. In order to approximate a smooth boundary and to accurately represent the shape of particles, it is necessary to use a large number of computational lattice points. In addition, when a particle moves,

* Corresponding author.
E-mail addresses: feng@unt.edu (Z.-G. Feng), stathis.michaelides@utsa.edu (E.E. Michaelides).

0045-7930/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compfluid.2008.04.013

Nomenclature

a	acceleration
e	velocity direction in LBM
f	force density
F	force
G	gravitational acceleration
I	moment of inertia tensor of a particle
m	mass
Ma	Mach number
n	LBM distribution function
p	pressure
r	radius of a spherical particle
Re	Reynolds number
S	area
t	time variable
u, U	velocity
v, V	volume
w	weight factor used in LBM
x	position coordinate

Greeks

ρ	density
Ω	entire domain
$\sum S_i$	region occupied by particles
τ	relaxation time
μ	dynamic viscosity
σ	shear stress tensor
Γ	boundary surrounding a solid
ω	angular velocity

Subscripts/superscripts

f	property related to fluid
i	index
n	value at the nth time step
s	property related to particle

Special

$\rightarrow$	vector
$\sim$	tensor

its computational boundary changes after each time step. This introduces fluctuations in the particle simulation shape and the forces that act on the particle, a procedure that limits the ability of LBM to solve particle–fluid interaction problems at high Reynolds numbers and low grid resolution. Several modifications to the bounce-back scheme have been proposed, but they are complicated to implement numerically and have rarely been used by the research community.

Peskin [4] first developed the IBM in order to model the flow of blood in the heart. This method uses a fixed Cartesian mesh for the fluid, which is composed of Eulerian nodes. However, for the solid boundaries, which are immersed in the fluid, the IBM uses a set of Lagrangian boundary points, which are advected according to the nature of the fluid–solid interactions. Höfler and Schwarzer [5] presented a finite-difference method for particle-laden flows by adding a constraint force into the Navier–Stokes equations to enforce the rigid motions of particles, with the constraint force being determined by a penalty method. Goldstein et al. [6] used the so-called "adaptive" or "feedback forcing scheme" to model the no-slip conditions on a stationary boundary. This technique necessitates the use of two free parameters that must be chosen, based on the flow conditions. ten Cate et al. [7] used a similar adaptive-forcing scheme with the LBM to simulate the sedimentation of a single sphere in an enclosure.

Feng and Michaelides [8] combined the IBM and the LBM by computing the force density using a penalty method in the simulations of particulate flows. In this hybrid, IB-LB method (IB-LBM), the particle boundary is treated as a deformable medium with a high stiffness. Thus, a small distortion of the particle boundary yields a restoring force that restores the particle to its original shape. These restoring forces exerted on the particle surface are distributed into the Eulerian nodes of the numerical grid and the flow field. The solution of the flow velocity over the whole fluid–particle domain is obtained by using the LBM. The IB-LBM is convenient to use, but has the disadvantage that it requires the *a priori* selection of the stiffness parameter based on the specific problem to be solved. In a later paper, Feng and Michaelides [9] addressed this issues by proposing a new computational method called *Proteus*, which employs the direct-forcing scheme for the computation of the force density functions. This direct-forcing scheme was originally proposed by Mohd-Yusof [10] for fixed complex boundaries and it eliminates the need for the determination of the free parameter for the stiffness coefficient. Thus, it makes the numerical method much more straightforward and efficient to use. Recently, the direct-forcing scheme has also been applied for solving particle–fluid heat transfer in particle laden flows [11].

Beside the treatment of no-slip boundary condition in LBM, another issue that has not been fully resolved is how to achieve numerical stability while updating the particle motion when the particle to fluid density ratio is close to or below unity. The scheme proposed by Ladd [1,2] exhibits instabilities when the particle to fluid density ratio is below 2, which imposes severe constrains on the application of LBM to light particles. Various schemes to resolve this issue have also shown to have their own limitations [12]. In this paper, we present a simple yet effective updating procedure that is capable to treat light particles in the flow. The procedure works even if the particle density is lower than the fluid.

In this paper, we briefly introduce the IBM-based scheme for treating the no-slip boundary conditions, provide two approaches for generating the enforcing points and present a stable and efficient scheme for updating the particle motion. We conduct a series of numerical simulations to study several issues associated with the IBM-based numerical procedure, such as the number of the enforcing points needed to accurately simulate a solid particle and the advantages and disadvantages of the method with respect to BBM. We validate the accuracy to the proposed method by comparison with other results and experimental data and, finally we demonstrate the effectiveness of the proposed updating velocity scheme by performing computations on the ascent of five light particles in a denser fluid. It must be emphasized that, in this paper, the IBM is introduced to enforce the no-slip condition. We do not attempt and we do not know if it is possible to the use this method for treating particles with a slip boundary condition.

2. Description of the numerical scheme

2.1. Simulation of fluid motion

We consider a particulate flow system composed of several rigid particles suspended in an incompressible Newtonian fluid, as shown in Fig. 1.

The entire computational domain, Ω, is composed by the fluid region, L, and the solid particle region, $\sum S_i$ ($= S_1 + S_2$ in the figure). The domain is surrounded by a boundary, Γ, and the boundary/

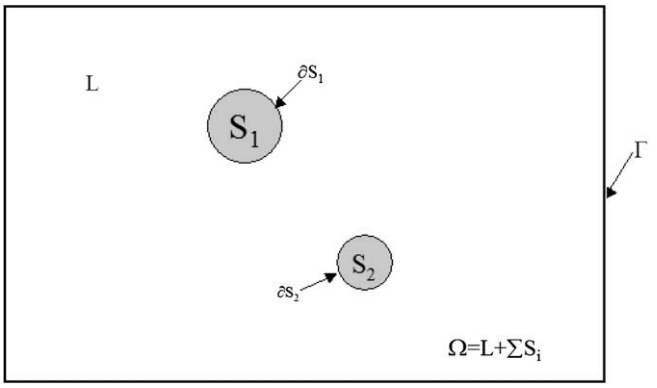

Fig. 1. Conceptual model of two circular particles suspended in a fluid.

surface of the ith particle S_i is denoted by ∂S_i. The particle region, S_i, is represented by the Lagrangian parametric coordinates, denoted by the symbol $\boldsymbol{s}$, and the flow domain, Ω, is represented by the Eulerian coordinates $\boldsymbol{x}$. Hence, any position on the particle region may be written as $\boldsymbol{x} = \boldsymbol{X}(\boldsymbol{s}, t)$. Let $f(\boldsymbol{x}, t)$ represent the fluid body force density. The governing equations for the fluid–particle composite domain are as follows:

$$\rho_f\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = \mu_f\nabla^2\boldsymbol{u} - \nabla p + \boldsymbol{f}, \tag{1}$$

$$\nabla\cdot\boldsymbol{u} = 0, \tag{2}$$

where ρ_f, μ_f and $\boldsymbol{u}$ are the fluid density, viscosity and velocity, respectively. The force density field $f(\boldsymbol{x}, t)$ is equal to zero in the regions occupied by the fluid. It is generally accepted that the interior field of the solid particles plays an insignificant role on the motion of the particles itself [12]. Therefore, the force density may be assumed to be zero in the interior of the solid particles. Then the force density is only prescribed over the surface region of particles and may be computed efficiently using a direct-forcing scheme.

In the numerical implementation of the method, the particle region is composed by a series of Lagrangian "enforcing points". Assuming that the velocity and pressure fields at the time step $t = t_n$ are known, we propose an explicit scheme to determine the force term at these enforcing points at time $t = t_{n+1}$, which is as follows:

$$f_i^{(n+1)} = \rho_f\left(\frac{u_i^{(n+1)} - u_i^{(n)}}{\delta t} + u_j^{(n)}u_{j,i}^{(n)}\right) - \mu_f u_{i,jj}^{(n)} + p_{,i}^{(n)}. \tag{3}$$

In order to impose the rigid-body motion condition that at $t = t_{n+1}$, the velocity on the enforcing points is equal to the velocity of the particle, $U_i^{S(n+1)}$, the force density at these points is given by the following expression:

$$f_i^{(n+1)} = \rho_f\left(\frac{U_i^{S(n+1)} - u_i^{(n)}}{\delta t} + u_j^{(n)}u_{j,i}^{(n)}\right) - \mu_f u_{i,jj}^{(n)} + p_{,i}^{(n)}. \tag{4}$$

The above equation is called "direct forcing", and was first introduced by Mohd-Yusof [10] for the treatment of fixed boundaries. The velocity at a Lagrangian points $\boldsymbol{x}$ is determined by the rigid-body motion of the particle:

$$\vec{u} = \vec{U}_s + \vec{\omega}_s \times (\vec{x} - \vec{x}_s), \tag{5}$$

where $\boldsymbol{U}_s$, $\boldsymbol{\omega}_s$ and $\boldsymbol{x}_s$ are the translational velocity, the angular velocity and the center of mass of the particle, respectively.

To solve the fluid field with a body force density, $f(\mathbf{x}, t)$, the LBM equation for the distribution function n_i is modified by an additional force term [11]

$$n_i(\boldsymbol{x} + \boldsymbol{e}_i, t+1) = n_i(\boldsymbol{x}, t) - \frac{1}{\tau}\left[n_i(\boldsymbol{x}, t) - n_i^{(0)}(\boldsymbol{x}, t)\right] + f_i^*(\boldsymbol{x}, t). \tag{6}$$

The additional forcing term f_i^* is obtained by the following first-order closure equation:

$$f_i^* = \frac{3}{2}w_i(\boldsymbol{e}_i\cdot\boldsymbol{f}). \tag{7}$$

A second-order equation for the body force can be written as follows according to Martys [13]:

$$f_i^* = w_i[(\boldsymbol{e}_i - \boldsymbol{u})\cdot\boldsymbol{f} + 3(\boldsymbol{e}_i\cdot\boldsymbol{u})(\boldsymbol{e}_i\cdot\boldsymbol{f})], \tag{8}$$

where the fluid particle direction vectors $\boldsymbol{e}_i$ ($i = 0, 1, 2, \ldots, 14$) corresponding to the column vectors are as in the following matrix [14]:

$$E = \begin{bmatrix} 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \end{bmatrix} \tag{9}$$

and the weight factors are given by the expressions

$$\begin{aligned} w_0 &= \frac{2}{9}, \\ w_i &= \frac{1}{9}, \quad i = 1, 2, \ldots, 6, \\ w_i &= \frac{1}{72}, \quad i = 7, 8, \ldots, 14. \end{aligned} \tag{10}$$

We compared the simulation results using the Eqs. (7) and (8), and found no significant difference; however, the use of Eq. (8) may slow down the computational performance by more than 15% in the cases we tested.

For the results described in this paper, a three-dimensional 15 bit (3DQ15) LBM model is used unless specified otherwise. As usual, the relaxation time, τ, is related to the kinematic viscosity of the fluid in the lattice unit by the expression

$$\nu = (2\tau - 1)/6. \tag{11}$$

The use of LBM for the approximation of an incompressible flow requires that the Mach number, Ma, be less than 0.3. One may show that Ma is related to the relaxation time τ as follows [8]:

$$Ma = \frac{\rho_f U_c \delta x}{\mu_f}\left(\frac{2\tau - 1}{2\sqrt{3}}\right), \tag{12}$$

where U_c and δx are the physical characteristic velocity and the grid step size. In the present simulation, τ is chosen such that the conditions for incompressible flow are satisfied.

2.2. Updating the motion of particles

The momentum interactions between the fluid and the particles cause the motion of particles, which includes both the translational motion and the rotational motion. In the original papers by Ladd [1,2], the proposed formula that was used for the translational velocity updating is as follows:

$$U_{n+1} = U_{n-1} + \frac{2F(t)\delta t}{m_{\text{eff}}}, \tag{13}$$

where m_{eff} is the effective mass of the particle, $m_{\text{eff}} = m_s - m_f$ (m_s and m_f are the mass of the particle and the mass of the fluid occupied by the volume of particle or internal fluid mass, respectively); $F(t)$ is the force acting on the particle; and δt is the time step. As pointed out later by Ladd and Verberg [12], there is a stability criterion that must be satisfied for this equation and its rotational counterpart:

$$\frac{\rho_s}{\rho_f} > 1 + \frac{10}{r}, \tag{14}$$

where r is the radius of the spherical particle in lattice unit. For most simulation cases, $r < 10$, and as a result, this requires a particle–fluid density ratio to be greater than 2. Other schemes have been proposed to resolve this instability problem, but they have been proven to be inefficient or to have other drawbacks [12].

The instability arises when the internal fluid mass term becomes comparable to the actual particle mass. As a result, the effective mass becomes small or even negative when the particle density is lower than the fluid density. To trace back the source of the instability, one might recall that the internal fluid mass term is due to the motion of particle. Unlike the gravity force or other driving forces, the force induced by the internal fluid mass is a passive force, that is, it is absent when the particle is not accelerating or decelerating. This force is a result of the particle acceleration, and, numerically, it should be determined by the motion of the particle during the previous time step. Therefore, we may separate the internal fluid mass from the actual mass term.

Let us consider the equation of the translational motion for the particle in its original form

$$m_s \frac{d\vec{U}_s}{dt} = \oint_{\partial S} \tilde{\sigma} \cdot d\vec{s} + (\rho_s - \rho_f) V_s \vec{g}, \tag{15}$$

where, ρ_s and V_s are the particle density and volume, respectively, and σ is the fluid stress tensor. The first term on the right-hand side of Eq. (15) is the interaction force between the particle and the surrounding fluid and the second term is the buoyancy force. The following surface integral yields the internal fluid mass term:

$$\oint_{\partial S} \tilde{\sigma} \cdot d\vec{s} = \int_S \vec{f}\, dv + \rho_f V_s \frac{d\vec{U}_s}{dt}. \tag{16}$$

Substituting Eq. (16) into Eq. (15) yields

$$m_s \frac{d\vec{U}_s}{dt} = \int_S \vec{f}\, dv + (\rho_s - \rho_f) V_s \vec{g} + m_f \frac{d\vec{U}_s}{dt}, \tag{17}$$

In Eq. (17), we deliberately did not combine the internal fluid mass term with the actual mass term, knowing that this term is the result of particle acceleration. Accordingly, the internal fluid mass term at time step $t = t_n + 1$ should be evaluated during the previous step, $t = t_n$, in the numerical implementation. In order to achieve this, we discretized Eq. (17) as follows:

$$m_s \frac{\vec{U}_s^{n+1} - \vec{U}_s^n}{\delta t} = \sum_i \vec{f}_i^n \delta V_i + (m_s - m_f)\vec{g} + m_f \vec{a}_s^n, \tag{18}$$

where $\boldsymbol{f}_i$ is the force density at the Lagrangian enforcing points, $\boldsymbol{x}_i$; δV_i is the volume associated with this enforcing point; the summation is taken on the whole particle region; and $\boldsymbol{a}_s$ is the translational acceleration of the particle, which may be evaluated explicitly as follows:

$$\vec{a}_s^n = \frac{\vec{U}_s^n - \vec{U}_s^{n-1}}{\delta t}. \tag{19}$$

Upon substituting the above equation into (18), one obtains the following update expression for the translational velocity:

$$\vec{U}_s^{n+1} = \left(1 + \frac{\rho_f}{\rho_s}\right)\vec{U}_s^n - \frac{\rho_f}{\rho_s}\vec{U}_s^{n-1} + \frac{\left(\sum_i \vec{f}_i^n \delta V_i + (m_s - m_f)\vec{g}\right)\delta t}{m_s}. \tag{20}$$

Similarly, for the rotational motion of a sphere, the original equation of motion is

$$\tilde{I}_s \frac{d\vec{\omega}_s}{dt} = \oint_{\partial S} (\vec{x} - \vec{x}_s) \times (\tilde{\sigma} \cdot d\vec{s}). \tag{21}$$

In the above equation, $\boldsymbol{I}_s$ is the particle's moment of inertia tensor. The surface integral will yield a "internal fluid moment of inertia tensor". In the numerical implementation, we separated the latter from the actual moment of inertia tensor. Finally, the following equation is obtained for updating the angular velocity:

$$\vec{\omega}_s^{n+1} = \left(1 + \frac{\rho_f}{\rho_s}\right)\vec{\omega}_s^n - \frac{\rho_f}{\rho_s}\vec{\omega}_s^{n-1} + \tilde{I}_s^{-1} \sum_i (\vec{x}_i - \vec{x}_s) \times \vec{f}_i^n \delta V_i. \tag{22}$$

The updating scheme described here works well even when the particle density is less than the density of the fluid, as it will be demonstrated in the numerical results of the following section.

2.3. Distribution of boundary nodes on the surface of a spherical particle

The direct-forcing method combined with LBM will be implemented in three-dimensional particle–fluid suspensions. The first issue arising is how to set up the Lagrangian enforcing points in order to represent adequately the particle region. As seen in Eqs. (20) and (22), the summation must be performed on the entire region occupied by the solid particles. However, since in most cases the fluid in the interior of the solids has insignificant contributions [12], an approximation is used and the summation of both Eqs. (20) and (22) is only evaluated at the solid surface region. This is similar to the BBM in which we simulate the motion of a spherical shell. The thickness of the shell is Δx, equal to the grid step size in the fixed grid. In the current implementation, the forcing points are assigned at the volume-based midway of the spherical shell. For a particle of radius r, the forcing points reside on a spherical surface of radius $r_b = \sqrt[3]{\frac{r^3 + (r - \Delta x)^3}{2}}$. When the particle radius r is large or the grid is very fine, the difference between assigning forcing points on

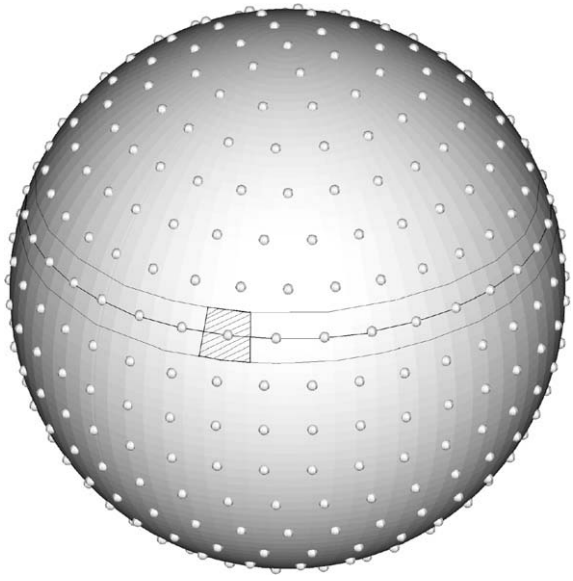

Fig. 2. Four hundred and sixteen enforcing points on a spherical surface of $r_b = 6$ using the strip approach.

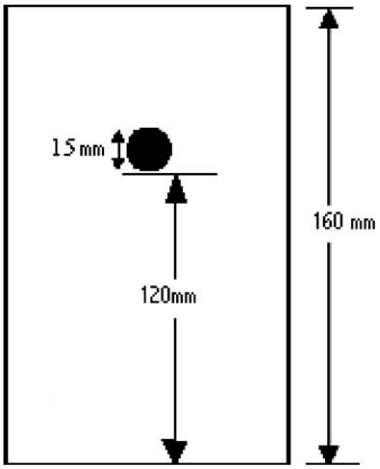

Fig. 3. Schematic diagram for a single particle settling in an enclosure.

Table 1
Fluid properties in the experiment and parameters used in simulations

	ρ_f (kg/m^3)	μ_f ($\times 10^{-3}$N s/m^2)	Re	τ	Δt ($\times 10^{-4}$s)
Case E1	970	373	1.5	0.9	3.47
Case E2	965	212	4.1	0.9	6.07
Case E3	962	113	11.6	0.75–0.8	8.51
Case E4	960	58	32.2	0.65	8.28

the surface of the particle or on a surface that is laid midway of this spherical shell is not significant.

For a spherical particle in three dimensions, it is impossible to find evenly distributed boundary nodes to represent the spherical surface. For this reason, Feng and Michaelides [9] used a number of strips to represent the surface of the sphere. The width of each strip is comparable to the grid spacing and each strip is composed by a number of evenly distributed points. The number of these points is chosen in a way that the spacing between two neighboring points is approximately equal to the width of the strips. The details of setting up the surface may be briefly described as follows: on the spherical surface of radius r_b, we construct $n_s = [\pi r_b + 1]$ lines to generate n_s strips with $k = 1$ and $k = n_s$ designating the north and south poles, which are associated by two half strips. For the kth ($k \neq 1$ and $k \neq n_s$) line, we assign $n_t = [2n_s \sin(k\pi/n_s)]$ points. The volume associated with each point in this strip is $dV = dS\Delta x$, where $dS = 4\pi r_b^2(\pi/n_s)\sin(k\pi/n_s)/n_t$. For the two special points of the north and south poles ($k = 1$ and $k = n_s$), the area is assigned to be equal to: $\pi^2 r_b^2/(n_s^2)$. For example, in the case of $r_b = 6$, which corresponds to an actual radius of spherical particle $r = 6.746$, a total of 416 points are assigned on the surface, as shown in Fig. 2. The area numerically added by these strips is equal to be 386.88, which is very close to the exact area of the spherical surface, 386.39. The simulation results indicate that the enforcing number proposed here is very conservative, it can even be cut to one-fourth without significantly affecting the computational results, as discussed in the following section.

Other approaches for generating enforcing points are also feasible. For example, it is well known that to get approximately evenly distributed points, one can let these points be charged particles, and apply a numerical simulation technique that minimizes the sum of the repulsive energy of the system of these particles. This approach has been used successfully by Uhlmann [15]. While it is expected that this approach would generate more evenly distributed points, since the volume assigned to each particle is taken as the average volume $dV = 4\pi r_b^2\Delta x/N$ (N is the number of the total enforcing points), any unevenness on the distribution of the enforcing points will result in an error on the assigned volume. It must be pointed out that in the strip approach we propose, the control volume associated with the enforcing points does not depend at all on the homogeneity of the distribution of these points, which are computed individually and independently.

3. Simulation results and discussions

3.1. Simulation of a sphere settling in an enclosure

3.1.1. Comparison with experiment data

The validation of the numerical schemes presented here has been accomplished by comparing the results obtained for spherical

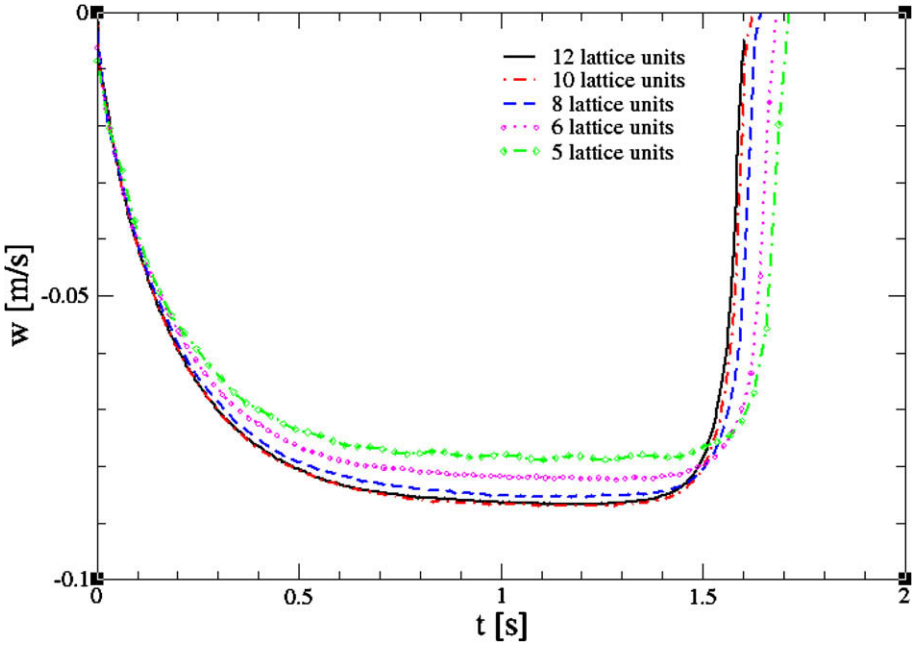

Fig. 4. Particle settling velocity for $Re = 11.6$ at various grids.

particles settling in an enclosure, with the experimental measurements by ten Cate et al. [7], who considered a spherical particle settling in a rectangular box of dimensions $10 \times 10 \times 16\ \text{cm}^3$ and made detailed measurements using a PIV system. The sphere commences its motion at a height $H = 12$ cm from the bottom, as depicted in Fig. 3. The fluid density was in the range from 960 to 970 kg/m^3 and its dynamic viscosity from 0.058 to 0.353 N s/m^2. The diameter of the solid particle was 150 mm and its density 1120 kg/m^3. Table 1 lists the fluid properties used in these experiments, the corresponding Reynolds numbers based on the measured maximum velocities and the typical numerical parameters used in our simulations.

For the convergence study and grid test, we chose the case E3 with $Re = 11.6$. We used a variable number of grids that makes the particle diameter in the range $d = 12$ to $d = 5$ lattice units. The instantaneous settling velocities of a single particle obtained from the different grids are shown in Fig. 4. It is apparent that the cases $d = 12$ and $d = 10$ yield almost identical results, an indication that representing the particle diameter with 10 grid points is adequate for obtaining sufficiently accurate results. The results of using a grid with $d = 8$, are also acceptable, but when the grid is very coarse with only a few grid points to outline the diameter of the particle, the results deviate significantly. We also tested two different relaxation times: $\tau_1 = 0.9$ and $\tau_2 = 0.75$ while keeping

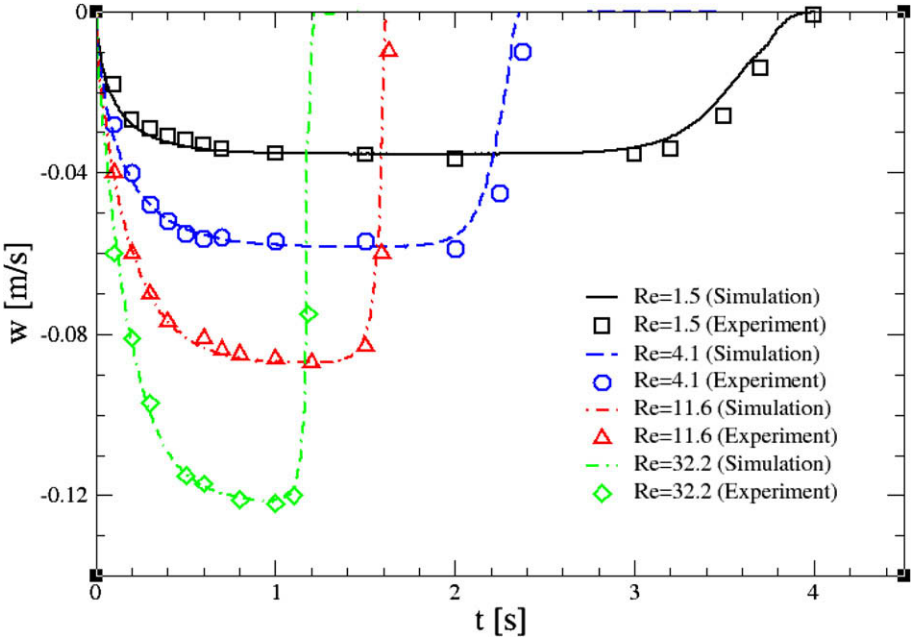

Fig. 5. Comparison of particle velocity with experimental data.

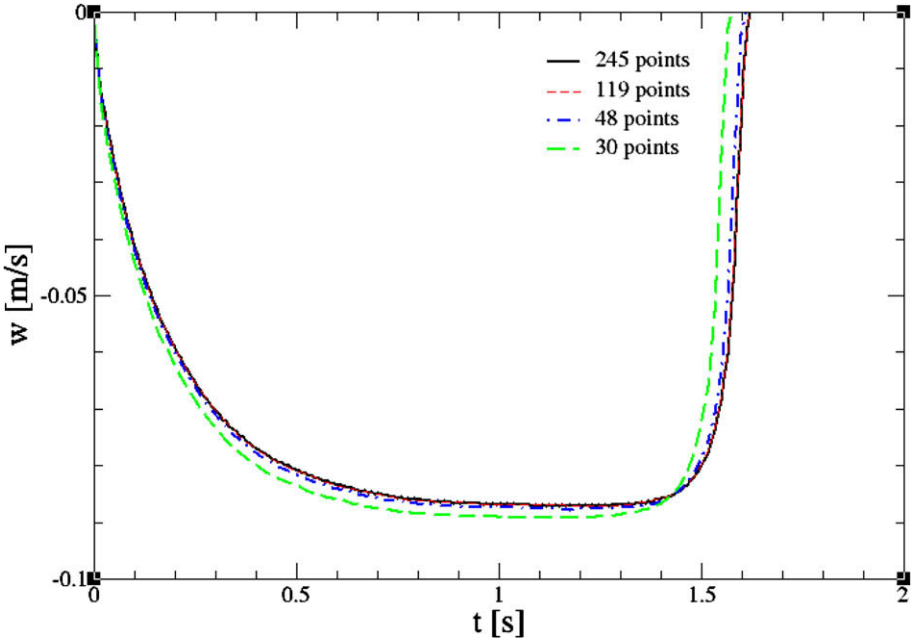

Fig. 6. Particle settling velocity at $Re = 11.6$ when different enforcing number is used.

the Reynolds number of the physical system unchanged, when the grid with $d = 12$ is used, and found out that they produce almost identical results.

Fig. 5 shows the particle instantaneous settling velocities obtained from the experimental measurements of ten Cate et al. [7] and from our numerical results for four different values of *Re*. A grid of $d = 12$ is used for all the four cases. It is observed that the simulation results for the particle velocity agree very well with the experimental measurements in all four cases. It must be pointed out that in Feng and Michaelides [9], the internal fluid mass term was neglected and this resulted in a difference of the order of 5% between the experimental measurements and the computational results.

3.1.2. Choosing the number of enforcing points

In this section, we investigate how the results will be affected when using different numbers of enforcing points. For the computations, we selected the case E3 of ten Cate et al. [7], where $Re = 11.6$. The particle diameter is 10 lattice units ($d = 10$); the enforcing points are located on the surface of a sphere with radius $r_b = 4.555$. When the strip scheme is used as described above, the number of strips is $n_t = 15$. The numbers of points on each strip are as follows: 1 (pole), 6, 12, 17, 21, 25, 27, 27, 27, 25, 21, 17, 12, 6, 1 (pole), for a total of $N = 245$ points. For the determination of the effect of the number of enforcing points we used the following cases:

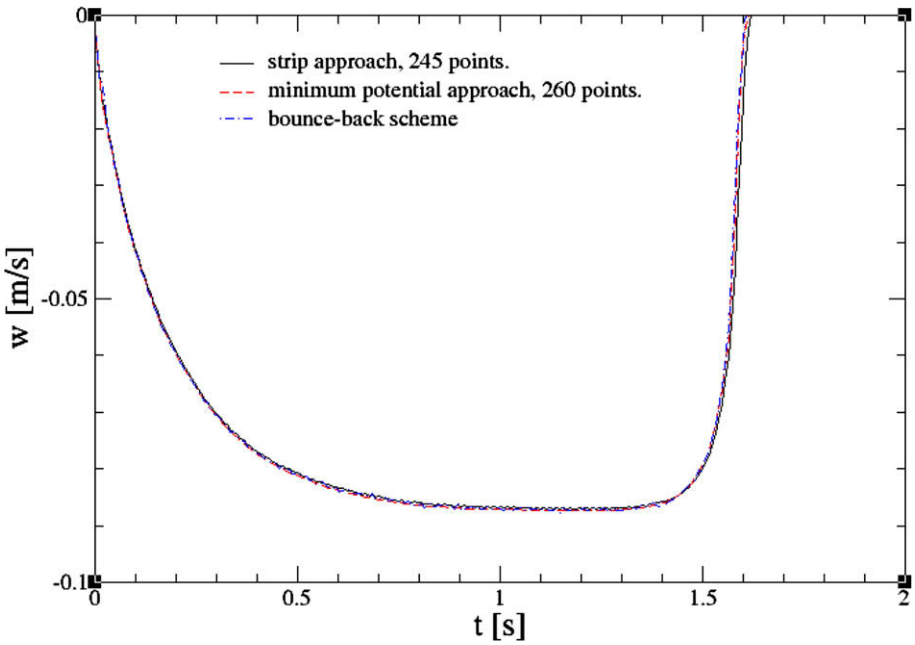

Fig. 7. Particle settling velocity at $Re = 11.6$ using different approaches with $d = 10$.

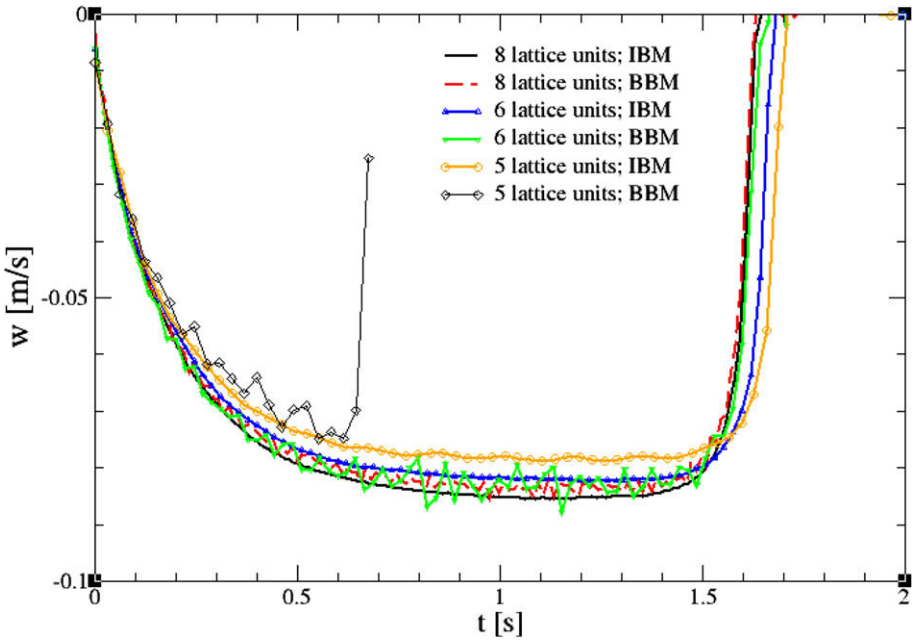

Fig. 8. Comparison of the IBM and BBM when different grid sizes were used.

Case A1: We set $n_s = [\pi r_b]$, and $n_t = [n_s^* \sin(k^*\pi/n_s)]$, resulting in $N = 119$ enforcing points on 14 strips, distributed as follows: 1, 3, 6, 8, 12, 13, 13, 13, 12, 10, 8, 6, 3, 1.

Case A2: We set $n_s = [0.8\pi r_b + 1]$, and $n_t = [n_s^* \sin(k^*\pi/n_s)]$, resulting $N = 74$ enforcing points on 12 strips, distributed as follows: 1, 3, 5, 8, 10, 10, 10, 10, 8, 5, 3, 1.

Case A3: We set $n_s = [0.65\pi r_b + 1]$, and $n_t = [n_s^* \sin(k^*\pi/n_s)]$, resulting $N = 48$ enforcing points on 10 strips, distributed as follows: 1, 3, 5, 7, 8, 8, 7, 5, 3, 1.

Case A4: We set $n_s = [0.5\pi r_b + 1]$, and $n_t = [n_s^* \sin(k^*\pi/n_s)]$, resulting $N = 30$ enforcing points on 8 strips, distributed as follows: 1, 3, 5, 6, 6, 5, 3, 1.

The simulation results together with the original case of $N = 245$ points are plotted in Fig. 6. It is apparent that there is very little difference in all the cases where the enforcing number is above 48. In the case of $N = 48$, the average volume associated by an enforcing point is equal to $4\pi r_b^2 \Delta x/N = 5.3$, which is significantly larger than one unit of volume. Nevertheless, the results for this case are considered very accurate.

We also used the minimum potential approach for generating enforcing points and found out that the computations produced identical results with our method. Fig. 7 shows the settling velocities obtained by using the strip-generation approach with a total 245 points, the minimum potential approach with a total forcing

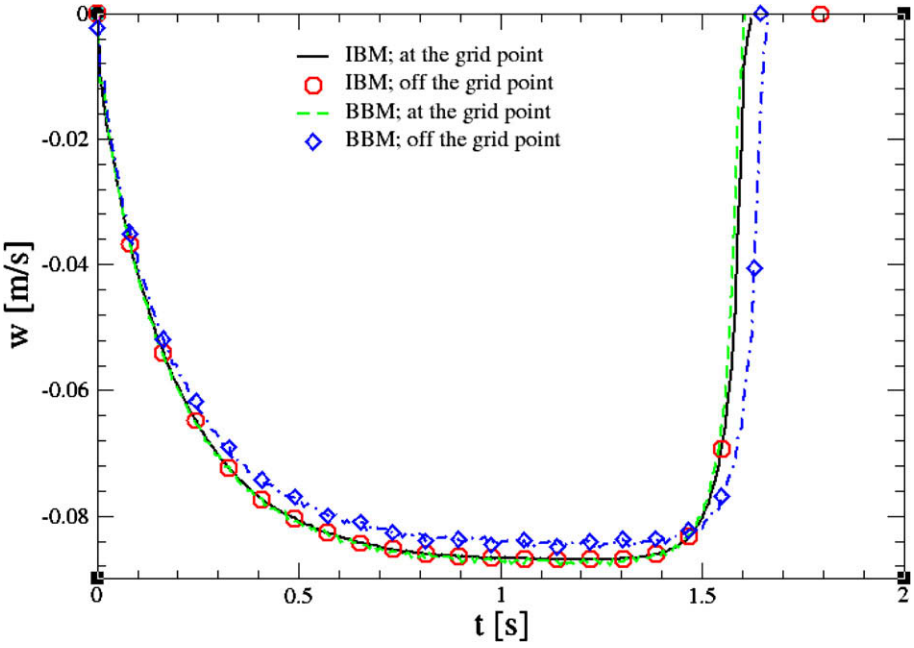

Fig. 9. Particle settling velocity at $Re = 11.6$ when original particle center is specified differently using IBM and BBM.

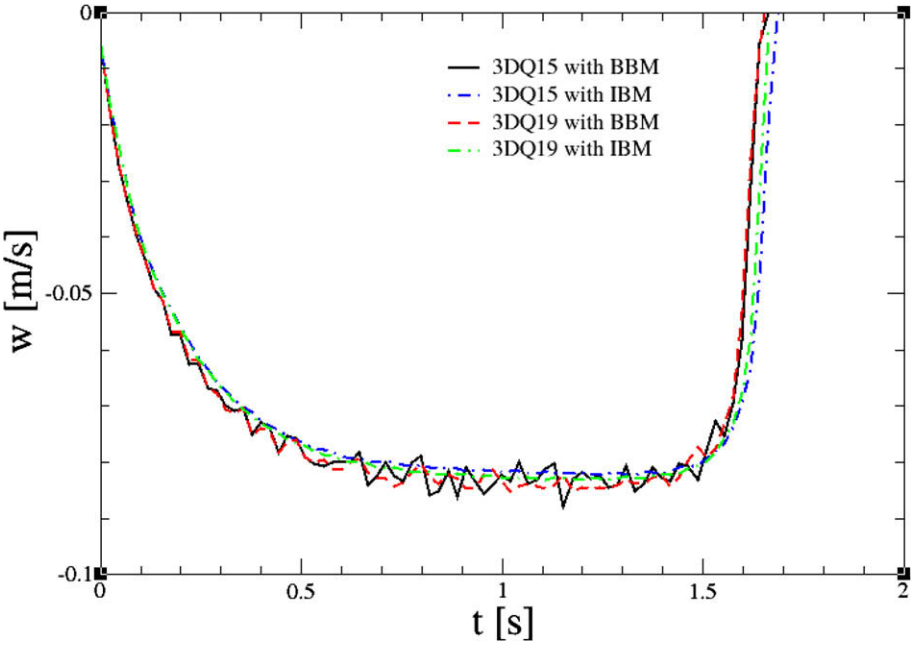

Fig. 10. Particle settling velocity at $Re = 11.6$ while using different simulation models.

number $N = [4\pi r_b^2] = 260$ and the classical bounce-back scheme. It is apparent that the results for the instantaneous settling velocity from these three methods show an excellent agreement.

3.1.3. Comparison to bounce-back method

In this section, we compare the results of using the IBM with results from BBM for treating the solid boundary by considering the case E3 with $Re = 11.6$. From the results depicted in Fig. 7, one may conclude that both the BBM and IBM yield reasonably good results at $d = 10$. However, as seen in Fig. 8, when the grid becomes coarser, the results from the BBM deteriorate significantly and exhibit fluctuations. In addition, we observed that, at $d = 5$, the simulations became unstable and the computation finally broke down. The IBM performed better in these situations and provided reasonable

Table 2
CPU time in second of running 100 time steps for the four different schemes used

	D3Q15 BBM	D3Q15 IBM	D3Q19 BBM	D3Q19 IBM
$54 \times 54 \times 85$ grid and 150 Lagrangian node	15	19	20	25
$120 \times 120 \times 192$ grid and 849 Lagrangian nodes	178	207	215	256

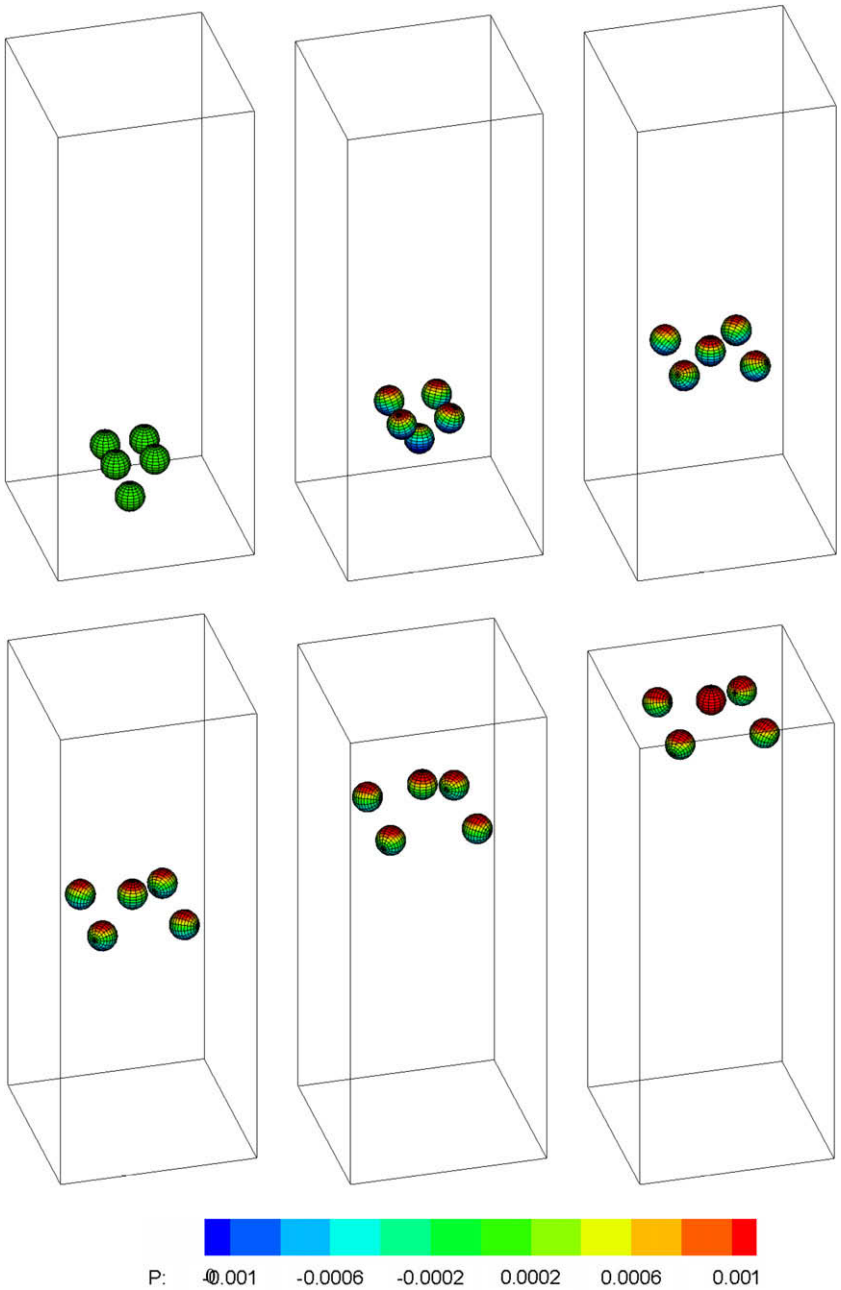

Fig. 11. Snapshots of particulate ascent during the time period $t = 0$, 0.5, 1, 1.5, 2.5 and 3.5 s with a time interval of 0.5 s. The color band shows the pressure contour.

results: No significant fluctuations were observed and the computations were concluded when the particle came to rest at the bottom of the container.

One of the main problems associated with the BBM is the imposed fluctuations on the shape of the particles at different time steps. This causes significant differences in the numerical simulation of the same particle at different instances and result in computational fluctuations. To further investigate the effect of the fluctuations of the boundary, we conducted a simulation using $d = 10$ and considered two cases: one case by putting the original particle center exactly at the lattice site, which lies along the intersection of the central x and y planes. The other is one-half Δx off both the central x and y planes. The small offset should have had a small effect to the actual settling velocity of particle, and this was confirmed by our simulation results using the IBM, whose results are depicted in Fig. 9. However, when we used the BBM we observed a significant difference in the computational results as shown in Fig. 9. One may conclude that the adoption of the IBM eliminates the boundary-induced fluctuations, which affect the computational results significantly. This enables one to use coarser grids for the simulation of particulate flow.

The above simulations are obtained using the 3DQ15 LBM model. Another commonly used LBM model for three-dimensional systems is the 3DQ19, where the fluid particles may move in 19

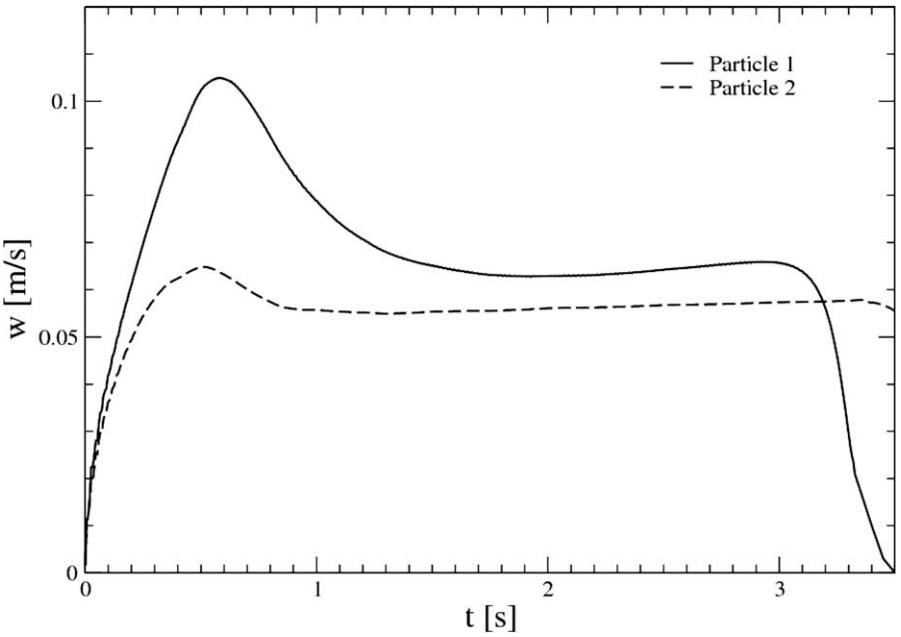

Fig. 12. Vertical ascent velocities for particle 1 and particle 2.

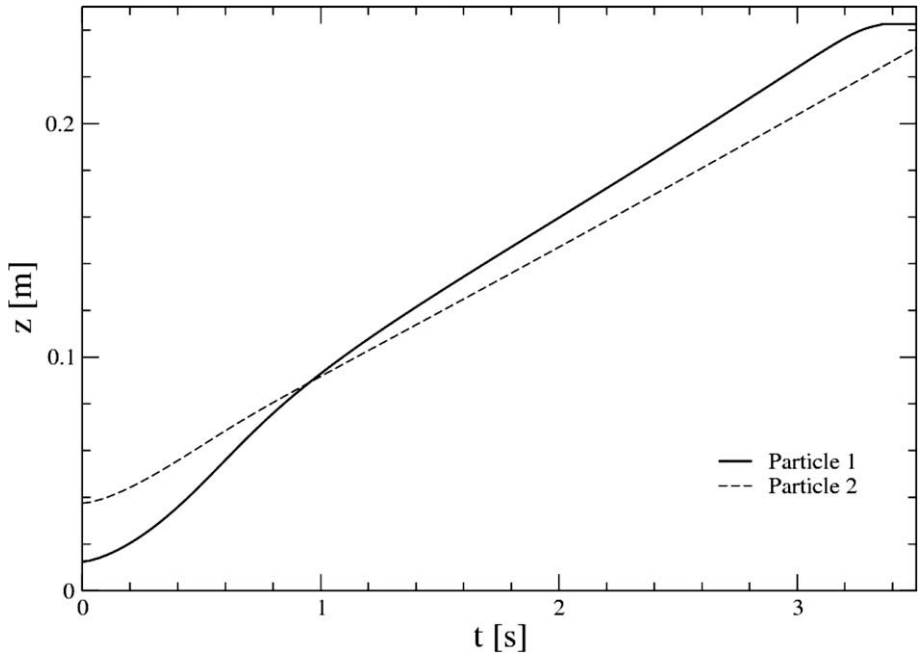

Fig. 13. Vertical positions of particle 1 and particle 2.

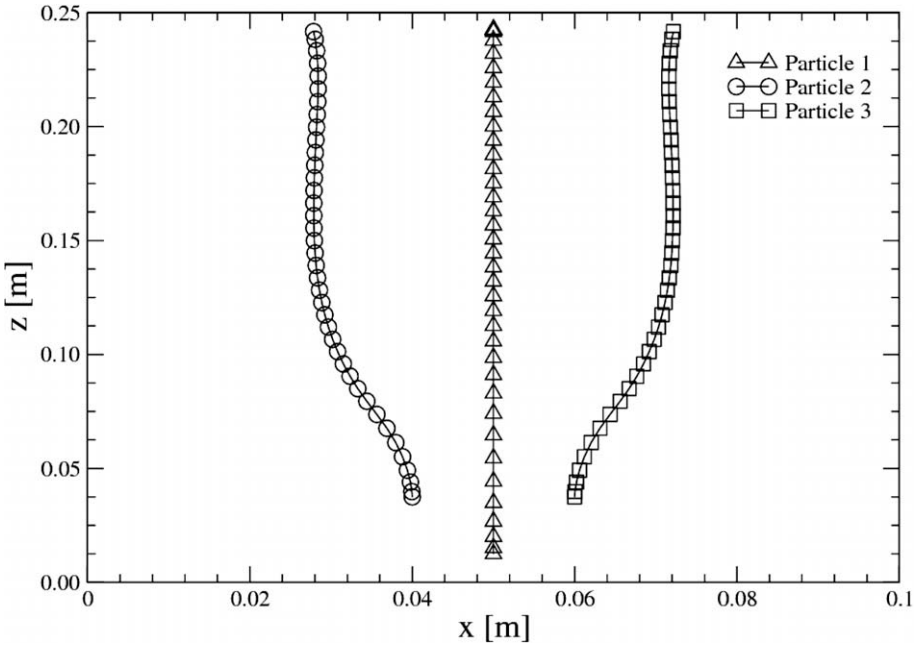

Fig. 14. Trajectories in x–z plane of the first three particles.

directions stemming from their center. We have implemented this model and applied it to simulate case E3. It is expected that 3DQ19 model is more accurate compared to 3DQ15 model, though it takes more computational time. Fig. 10 depicts the settling velocity of the particle in the case E3 using the four methods: 3DQ15 with LBM, 3DQ15 with IBM, 3DQ19 with LBM, and 3DQ19 with IBM. A grid of $40 \times 40 \times 64$ which makes the diameter of sphere to be equal to 6, and a total of 79 Lagrange points assigned on the surface of sphere are used. We found that the improvement of using 3DQ19 is not significant.

IBM uses a set of Lagrange points and tracks their motion as a way to outline the surface of a particle. This causes the increase in the computational time. To compare the performance between IBM and BBM, we used two different grid setups to run the simulation for case E3, with $Re = 11.6$. The simulations were conducted on a dual core Xeon (3 GHz CPU and 4 GM RAM) workstation. Table 2 shows the CPU time for running100 time steps simulation. The table includes the use of 3DQ19 model with IBM and 3DQ19 with BBM as well. Eq. (7) is used for the forcing term. It is observed that the IBM takes approximately 10–20% longer than the BBM. However, the improved accuracy of the IBM more than compensates for this time. It is also observed that the D3Q19 method is about 20% or more costly in terms of computational time, a cost may not be justified considering its improvement on the accuracy is very limited.

3.2. Rising of light particles

As pointed out above, the updating velocity scheme proposed in this paper is suitable even for particles with density lower than the fluid density. To demonstrate this capability, we considered the motion of five light particles in an enclosure. The enclosure has dimensions $0.1 \times 0.1 \times 0.25$ m. In the simulations, the particle diameter is $d = 0.015$ m; the fluid viscosity is 0.0002 N s/kg; and the particle to fluid density ratio is 0.8. The initial ($t = 0$) particle positions are as follows:

Particle 1 (the bottom particle seen in the first snapshot of Fig. 11): (0.05, 0.05, 0.0125).

Particle 2–5: (0.04, 0.04, 0.0375), (0.06, 0.04, 0.0375), (0.04, 0.06, 0.0375), (0.06, 0.06, 0.0375).

An $80 \times 80 \times 200$ grid is used for the simulations, which makes the particle diameter equal to 12 lattice units. The relaxation time $\tau = 0.75$ was used, which gives a physical time step of 0.000651 s. The maximum particle diameter based Reynolds number encountered in the simulation is approximately 8. Fig. 11 shows the snapshots of these particles during the simulation time 0–3.5 s. The color on the particles indicates the pressure contours on the particle surface: the particle top regions have higher pressure, as expected. Particle rotation is also shown by the orientation of the mesh.

Figs. 12–14 show the particle vertical rising velocity, position and trajectory. Due to the symmetric arrangement of particles 2–5, the rising velocities and positions in the vertical direction for these particles are the same. From the simulation results, we can see that the five particles start rising, due to the strong buoyancy force and the leading four particles create wakes behind them. As a result, the trailing particle acquires sufficient momentum to overtake them and push them aside. This causes small components of lateral motion and rotation on the last four particles. As early as $t = 0.9$, the first particle passes the other four particles and at $t = 3.5$, this particle starts to slow down due to the deceleration caused by the presence of the top wall.

4. Conclusions

This paper examines two problems related to the simulations using the LBM for particle–fluid suspensions: the first is the treatment of the no-slip boundary condition on the particle surface and the second is the updating of the velocity for light particles. The use of the bounce-back scheme for the no-slip boundary condition has shown to generate significant fluctuations when the grid is not sufficiently fine. This problem is avoided by applying the immersed boundary scheme in the computations. We have discussed the several parameters and issues associated with this scheme and our simulation results have shown that it is robust and, under the same conditions, provides better results than the bounce-back scheme. However, IBM is more computational costly compared to BBM. The problem of updating the velocity for light particles has also been addressed satisfactorily by separating the internal fluid mass

of the particle from the actual particle mass. The proposed updating scheme has been proven very effective and provides results even for particles that are lighter than the fluid and ascend in the flow field.

References

[1] Ladd AJC. Numerical simulations of particulate suspensions via a discretized Boltzmann equation Part I. Theoretical foundation. J Fluid Mech 1994;271:285.
[2] Ladd AJC. Numerical simulations of particulate suspensions via a discretized Boltzmann equation Part II. Numerical results. J Fluid Mech 1994;271:311.
[3] Frisch U, Hasslacher B, Pomeau Y. Lattice–gas automata for the Navier–Stokes equations. Phys Rev Lett 1986;56:1505.
[4] Peskin CS. Numerical analysis of blood flow in the heart. J Comput Phys 1977;25:220.
[5] Höfler K, Schwarzer S. Navier–Stokes simulation with constraint forces: finite-difference method for particle-laden flows and complex geometries. Phys Rev E 2000;61:7146.
[6] Goldstein D, Handler R, Sirovich L. Modeling a no-slip flow boundary with an external force field. J Comput Phys 1993;105:354.
[7] ten Cate A, Nieuwstad CH, Derksen JJ, van den Akker HEA. Particle imaging velocimetry experiments and lattice-Boltzmann simulations on a single sphere settling under gravity. Phys Fluids 2002;14:4012.
[8] Feng Z-G, Michaelides EE. An immersed boundary method combined with lattice Boltzmann method for solving fluid and particles interaction problems. J Comput Phys 2004;195:602.
[9] Feng Z-G, Michaelides EE. Proteus: a direct forcing method in the simulations of particulate flows. J Comput Phys 2005;202:20.
[10] Mohd-Yusof J. Combined immersed boundaries/B-splines methods for simulations of flows in complex geometries, Annual Research Briefs, Center for Turbulence Research, Stanford University, 1997.
[11] Feng Z-G, Michaelides EE. Inclusion of heat transfer computations for particle laden flows. Phys Fluids 2008;20.
[12] Ladd AJC, Verberg R. Lattice-Boltzmann simulation of particle–fluid suspensions. J Statist Phys 2001;104:1191.
[13] Martys NS. Improved approximation of the Brinkman equation using a lattice Boltzmann method. Phys Fluids 2001;13:1807.
[14] Feng Z-G, Michaelides EE. Hydrodynamic force on spheres in cylindrical and prismatic enclosures. Int J Multiphase Flow 2002;28:479.
[15] Uhlmann M. An immersed boundary method with direct forcing for the simulation of particulate flows. J Comput Phys 2005;209:448.