Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2022-0058

Vol. **33**, No. 1, pp. 349-366
January 2023

Effective Force Stabilising Technique for the Immersed Boundary Method

Arnab Ghosh[1,*], Alessandro Gabbana[1], Herman Wijshoff[2,3] and Federico Toschi[1]

[1] *Department of Applied Physics and Science Education, Eindhoven University of Technology, PO Box 513, 5600 MB Eindhoven, The Netherlands.*
[2] *Department of Mechanical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.*
[3] *Canon Production Printing Netherlands B.V., P.O. Box 101, 5900 MA Venlo, The Netherlands.*

Received 21 February 2022; Accepted (in revised version) 5 November 2022

Abstract. The immersed boundary method has emerged as an efficient approach for the simulation of finite-sized solid particles in complex fluid flows. However, one of the well known shortcomings of the method is the limited support for the simulation of light particles, *i.e.* particles with a density lower than that of the surrounding fluid, both in terms of accuracy and numerical stability.
Although a broad literature exists, with several authors reporting different approaches for improving the stability of the method, most of these attempts introduce extra complexities and are very costly from a computational point of view.
In this work, we introduce an effective force stabilizing technique, allowing to extend the stability range of the method by filtering spurious oscillations arising when dealing with light-particles, pushing down the particle-to-fluid density ratio as low as 0.04. We thoroughly validate the method comparing with both experimental and numerical data available in literature.

AMS subject classifications: 65D10, 76M28

Key words: Immersed boundary method, lattice Boltzmann method, light particle, force stabilization, added mass effect.

1 Introduction

The transportation of rigid particles, droplets and bubbles in multiphase and multicomponent fluid flows are ubiquitous to several fields of science and technology [1]. Many

*Corresponding author. *Email addresses:* `a.ghosh1@tue.nl` (A. Ghosh), `a.gabbana@tue.nl` (A. Gabbana), `h.m.a.wijshoff@tue.nl` (H. Wijshoff), `f.toschi@tue.nl` (F. Toschi) examples can be cited, for example in connection with environmental research (erosion of sediments in sea shores and bubble generation in thee ocean), power engineering (to enhance heat and mass transfer inside of the boiler of power-plants), chemical engineering (reaction between a gas and a liquid phase relies on the increased surface area of small bubbles), bio-medical industry (air bubble formation in transported blood samples through pneumatic tube systems) and many others.

The motion of particulate matter, as well as the numerical techniques required for an accurate and reliable description of its dynamic, may vary significantly depending on whether one deals with bubbles, droplets or particles. For example, for heavy particles, with densities larger than that of the dispersed phase, the dynamics is mostly governed by the inertia of the particles. On the other hand, when dealing with light particles (e.g. small bubbles) the governing force comes mostly from the inertia of the fluid inside the immersed body which gets accelerated along with the particle; this phenomena is commonly referred as "added mass effect" (and seldom as "internal mass effect" [2]).

Although a significant effort has been invested through the years on the experimental side for the study of the motion of bubbles in complex fluid flows [3–5], there is not as much literature available in terms of numerical works, due to the lack of efficient techniques for the simulation of light particles in fluid flows.

A standard approach for the simulation of interactions between fluids and particles is given by the Immersed Boundary Method (IBM), which simulates the boundary of the particles using a Lagrangian grid. The method, originally introduced by Peskin [6, 7], and successively refined over the years by a number of researchers [8–11], has proven successful in the simulation of several types of complex fluid-particle interactions. However, a well known shortcoming of the method is the restricted support, both in terms of accuracy and numerical stability, for the simulation of light particles [10].

Although several methods have been reported for extending the stability range of IBM (e.g. [12–14]), they are often very expensive from a computational point of view, and for this reason one usually relies on other numerical approaches for the simulation of light particles, such as for example the interface tracking method [15, 16].

In this work, we present a lightweight solution for filtering-out spurious oscillations arising in the force term acting on the particle, which occur when simulating light particles in fluid flows.

We couple the IBM with a Lattice Boltzmann Method (LBM) [17] for the solution of the governing equations of the fluid, and perform numerical simulations for heavy and light particles, comparing and validating against both numerical and experimental data.

Our results show that we are able to solve particle to fluid density ratios as low as 0.04, improving of about one order of magnitude over a standard IBM implementation.

This article is organized as follows: in Section 2 we introduce the numerical methods used in the simulation of the fluid dynamics and of the fluid-particle interactions, respectively the LBM and the IBM. Besides, we also provide a description of the shortcoming of the IBM in the simulation of light particles, as well as a stabilization technique for smoothing out oscillations from the particle force and torque. In Section 3 we report numerical results from two different test-beds, evaluating accuracy and stability of the method. Finally, in Section 4 we summarize our findings, together with an outlook on possible future developments.

2 Methodology

2.1 Lattice Boltzmann Method

In this section, we give a short introduction to the Lattice Boltzmann Method (LBM), which we employ for solving the governing equations of the fluid.

The LBM (see e.g [17,18] for a detailed introduction) simulates the evolution of macroscopic quantities (such as density and velocity) through a mesoscopic approach based on the synthetic dynamics of a set of discrete velocity distribution functions $f_i(\boldsymbol{x},t)$, to which we will refer to as lattice populations.

At each grid node $\boldsymbol{x}$, the lattice populations are defined along the discrete components of the stencil $\{c_i\}$, $i=1,\cdots,q$. It is customary to distinguish between different LBMs using the DdQq nomenclature, in which d refers to the number of spatial dimensions and q to the number of discrete components. In this work we adopt the D3Q19 model (see [19] for the implications of the choice of the lattice structure), to implement the single-relaxation time lattice-BGK [20]:

$$f_i(\boldsymbol{x}+\boldsymbol{c}_i\Delta t,t+\Delta t)=f_i(\boldsymbol{x},t)-\frac{\Delta t}{\tau}\left(f_i(\boldsymbol{x},t)-f_i^{\text{eq}}(\boldsymbol{x},t)\right)+F_i(\boldsymbol{x},t). \tag{2.1}$$

In the above, Δt is the time step, τ the relaxation time, F_i a discrete force density term, and $f_i^{\text{eq}}(\boldsymbol{x},t)$ is the discrete equilibrium distribution, for which we use a second order Hermite-expansion of the Maxwell-Boltzmann distribution:

$$f_i^{\text{eq}}(\rho,\boldsymbol{u})=w_i\rho\left(1+\frac{\boldsymbol{u}\cdot\boldsymbol{c}_i}{c_s^2}+\frac{(\boldsymbol{u}\cdot\boldsymbol{c}_i)^2-(c_s|\boldsymbol{u}|)^2}{2c_s^4}\right),$$

with w_i a lattice-dependent set of weighting factors, $c_s=1/\sqrt{3}$ the sound speed in the lattice, and ρ and $\boldsymbol{u}$ respectively the density and the velocity field.

For the implementation of the forcing term F_i, we adopt Guo's model [21]:

$$F_i(\mathbf{x},t)=\left(1-\frac{1}{2\tau}\right)w_i\left[\frac{\mathbf{c}_i-\mathbf{u}(\mathbf{x},t)}{c_s^2}+\frac{\mathbf{c}_i\cdot\mathbf{u}(\mathbf{x},t)}{c_s^4}\mathbf{c}_i\right]\cdot\mathbf{F}(\mathbf{x},t), \tag{2.2}$$

where $\boldsymbol{F}(\boldsymbol{x},t)$ represents the force vector acting on the system.

The macroscopic observable ρ and $\boldsymbol{u}$, can be calculated from the moments of the velocity distribution functions:

$$\rho=\sum_{i=1}^{q}f_i,\quad \rho\mathbf{u}=\sum_{i=1}^{q}f_i\mathbf{c}_i+\frac{\Delta t}{2}\mathbf{F}(\boldsymbol{x},t), \tag{2.3}$$

with a correction in the first order moment due to Guo's model.

Following an asymptotic analysis, such as the Chapman-Enskog expansion, it can be shown that Eq. (2.1) provides a second order approximation of the Navier-Stokes equations, with the following expression putting in relationship the relaxation time parameter τ with the kinematic viscosity ν of the fluid:

$$\nu=\left(\tau-\frac{1}{2}\right)c_s^2. \tag{2.4}$$

We conclude this section by sketching the LBM algorithm, which, provided a suitable initialization of the particle distribution functions, consists at each iteration of the following steps:

1. Perform the streaming step:
$$f_i^*(\boldsymbol{x},t)=f_i(\boldsymbol{x}-\boldsymbol{c}_i\Delta t,t). \tag{2.5}$$
2. Coupling with the Immersed Boundary Method in order to i) apply boundary conditions, and ii) calculate $\boldsymbol{F}(\boldsymbol{x},t)$
3. Calculate the discrete force term via Eq. (2.2).
4. Compute the macroscopic fields using Eq. (2.3)
5. Apply the collisional operator
$$f_i(\boldsymbol{x},t+\Delta t)=f_i^*(\boldsymbol{x},t)-\frac{\Delta t}{\tau}\left(f_i^*(\boldsymbol{x},t)-f_i^{\text{eq}}(\boldsymbol{x},t)\right)+F_i(\boldsymbol{x},t). \tag{2.6}$$

2.2 Immersed boundary method

In this section we introduce the immersed boundary method (IBM) used to simulate the fluid particle interaction as well as complex moving boundaries. We follow the implementation of Uhlmann [10] which is based on a distribution of Lagrangian markers to denote the immersed boundary as shown in the Fig. 1. The fluid dynamics evolves on Eulerian nodes (i.e. a Cartesian grid) and the no-slip boundary condition are imposed at the boundaries represented by the Lagrangian markers. Starting from Eq. (2.5), the density and the (*un-forced*) velocity $\mathbf{u}^{noF}$ is obtained from Eq. (2.3). The non-forced velocity $\mathbf{u}^{noF}$ is then interpolated from the Eulerian nodes to the Lagrangian markers using the following equation:

$$\mathbf{u}_b^{noF}(\mathbf{x}_b)=\sum_{\mathbf{x}\in g_h}\mathbf{u}(\mathbf{x})D(\mathbf{x}-\mathbf{x}_b)(\Delta h)^3, \tag{2.7}$$

where summation runs over all the grid points of the Cartesian grid (g_h) used for solving the fluid equation, $\mathbf{x}_b$ represents the position of the Lagrangian markers, Δh represents the lattice size, $\mathbf{u}(\mathbf{x})$ is the fluid velocity on the Eulerian nodes and $\mathbf{u}_b^{noF}$ is the interpolated

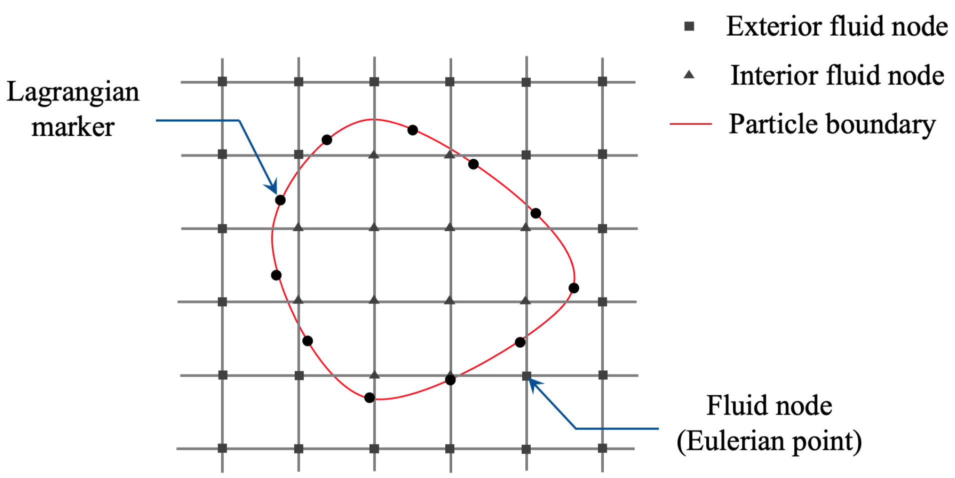


Figure 1: Schematic representation of a particle in the immersed boundary method. The straight horizontal/vertical lines represent the Cartesian grid and the intersections of the lines (black squares and triangles) represent the Eulerian (fluid) nodes where the time evolution of the fluid field is calculated. The black squares denote the fluid nodes which are outside the immersed boundary and the black triangles represent the fluid nodes which are inside the immersed boundary. A set of Lagrangian markers (black circles) are used to represent the particle boundary (red line). The effect of the interaction between the particle and the fluid is reconstructed at the fluid nodes via interpolation.

velocity on the immersed boundary. Moreover, $D(\mathbf{x}-\mathbf{x}_b)$ is the delta function used in the interpolation, defined as follows:

$$D(\mathbf{x}-\mathbf{x}_b)=\frac{1}{(\Delta h)^3}d_h\left(\frac{x-x_b}{\Delta h}\right)d_h\left(\frac{y-y_b}{\Delta h}\right)d_h\left(\frac{z-z_b}{\Delta h}\right), \tag{2.8}$$

with $d_h(r)$ given by:

$$d_h(r)=\begin{cases} 1-|r|, & |r|\leqslant 1, \\ 0, & |r|>1. \end{cases} \tag{2.9}$$

In the above, r represents the distance between the Eulerian node and the Lagrangian marker. Following Uhlmann [10], the force density $\mathbf{F}_b$ is calculated at the Lagrangian markers as:

$$\mathbf{F}_b=\rho\frac{\mathbf{u}_b^d-\mathbf{u}_b^{noF}}{\Delta t}, \tag{2.10}$$

where $\mathbf{u}_b^{noF}$ is the interpolated velocity and $\mathbf{u}_b^d$ the desired target velocity. The force density is then *spread* to the nearby Eulerian nodes based on the relation:

$$\mathbf{F}(\mathbf{x},t)=\sum_b \mathbf{F}_b D(\mathbf{x}-\mathbf{x}_b)\Delta S_b, \tag{2.11}$$

with ΔS_b the surface area of the immersed boundary.

Finally, the result of Eq. (2.11) is used by LBM to advance the time stepping of Eq. (2.1).

2.3 Particle dynamics

The particle dynamics is described by Newton's laws of motion, which we solve using the well know Leapfrog algorithm [22]. In particular, we need to determine at each time step the position, orientation, as well as the translational and angular velocities of the particle.

The translational force balance acting on the particle is given by

$$M_p \frac{d\mathbf{u}_p}{dt} = M_f \frac{d\mathbf{u}_p}{dt} - \sum_b \mathbf{F}_b \Delta V_p + (M_p - M_f)\mathbf{g}, \tag{2.12}$$

with M_p the mass of the particle, M_f the mass of the fluid enclosed within the particle volume ($M_f = \rho_f V_p$), $\mathbf{u}_p$ the particle velocity and $\mathbf{g}$ the acceleration due to gravity. The three terms on the right hand side of Eq. (2.12) represent, respectively, the added mass force, the hydrodynamic force and the gravity (and buoyancy) force acting on the particle.

Likewise, the torque balance acting on the particle can be expressed as:

$$I_p \frac{d\boldsymbol{\omega}_p}{dt} = I_f \frac{d\boldsymbol{\omega}_p}{dt} - \sum_b (\boldsymbol{x}_b - \boldsymbol{x}_c) \times \mathbf{F}_b \Delta V_p, \tag{2.13}$$

with I_p the moment of inertia of the particle, I_f the moment of inertia of the fluid enclosed within the particle volume, $\boldsymbol{x}_c$ the center of the particle and $\boldsymbol{\omega}_p$ the angular velocity of the particle.

The target velocity on the boundary Lagrangian markers ($\mathbf{u}^d$) can be calculated by combining the translational and angular velocities:

$$\mathbf{u}_b^d(\boldsymbol{x}_b) = \boldsymbol{u}_p + \boldsymbol{\omega}_p \times (\boldsymbol{x}_b - \boldsymbol{x}_c), \tag{2.14}$$

where $\boldsymbol{x}_c$ represents the center of the particle.

Finally, in order to determine the angular orientation of the particle, we employed the quaternions technique [22] (see Appendix A for more details).

2.4 Problem statement

In this section, we provide an overview of the issues arising when simulating light particles in fluid flows with the IBM implementation introduced by Uhlmann [10].

Hereafter, we use the term "light particles" to refer to particles with a particle to fluid density ratio less than unity ($\rho_p/\rho_f < 1$); conversely, we use "heavy particles" to reference to the case $\rho_p/\rho_f > 1$.

When considering light particles, the governing force comes from the inertia of the fluid enclosed within the immersed boundary of the particle, which gets itself accelerated with the particle. This phenomenon is known in the literature as the "added mass force". Numerical simulations under these settings yields strong oscillations in both velocities and force (and torque) [12, 13], which ultimately lead to numerical instabilities and/or poor agreement with experimental results. Formally, the translational motion of a particle is represented by the equation:

$$m_p \frac{d\mathbf{u}_p}{dt} = \frac{d}{dt}\int_{V_p} \rho_f \mathbf{u} dV - \rho_f \int_{V_p} \mathbf{f} dV + V_p(\rho_p - \rho_f)\mathbf{g}. \tag{2.15}$$

In the above, the first term on the right hand side represents the *added mass term*. The integration of such term needs to be performed over all fluid nodes enclosed within the immersed boundary, which in turn requires carefully chosen interpolation techniques [12] in order to reach good accuracy. Therefore, it is customary to replace its calculation with the so called rigid-body approximation [2]:

$$\frac{d}{dt}\int_{V_p} \rho_f \mathbf{u} dV \approx \rho_f V_p \frac{d\mathbf{u}_p}{dt}, \tag{2.16}$$

where the "added mass term" can be represented using the acceleration of the particle. However, even though the rigid-body approximation is more efficient from a computational point of view, it induces instabilities especially in the case of light particles [12]. On the other hand, explicitly performing the integration leads to stable and accurate simulations [12–14].

In the following, we provide an example of simulation of a light particle with $\rho_p/\rho_f = 0.7$, and the following parameters, expressed in Lattice Units (LU): we consider a domain surrounded by walls, with no slip boundary conditions, represented on a grid of size $32\times 32\times 64$, simulating a particle of radius of $R_p = 8$ LU. The Reynolds number of the particle, which is defined as $Re_p = 2R_p u_0/\nu$, is set to 0.01, with u_0 being the Stokes velocity of the particle, defined as

$$u_0 = \frac{2gR_p^2}{9\nu}\left(\frac{\rho_p}{\rho_f} - 1\right). \tag{2.17}$$

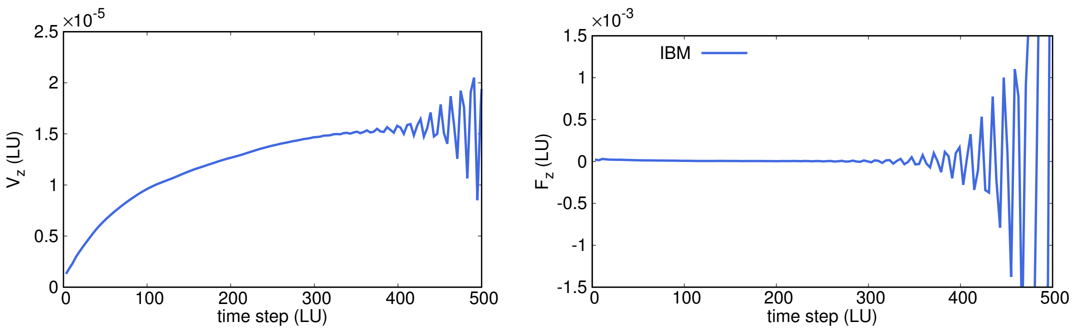
The gravity force acting on the system is $g = 1.757\times 10^{-6}$ LU, and the relaxation time, controlling the viscosity of the fluid, set to $\tau = 0.65$. This sets the viscosity of the fluid to $\nu = 0.05$ LU as for Eq. (2.4).

We employ the rigid-body approximation (Eq. (2.16)). In Fig. 2 we highlight the spurious oscillations arising after approximately 400 time steps in the vertical component of both velocity and force.

In the next section, we describe a simple technique for damping the observed oscillation.

2.5 Force stabilization

In this section, we introduce a simple approach for the simulation of light particles, which allows damping the oscillations described in the previous section, without the need of resorting to the rigid body approximation, which would require the computationally costly evaluation of the integral in Eq. (2.15).



(a) velocity over time steps (b) force over time steps

Figure 2: Instability in *(a)* the vertical force component, and in *(b)* the vertical velocity component of a particle with density lighter than the surrounding fluid ($\frac{\rho_p}{\rho_f}=0.7$) using the standard IBM implementation (blue line). The particle Reynolds number is set to $\mathrm{Re_p}=0.01$. The stable result represented by the red line is obtained applying a moving average to the force calculation (IBM+MA).

We start from Eq. (2.12) and Eq. (2.13), giving respectively the total force and torque acting on the particle. These quantities are calculated at each time step and used to define the particle dynamics by solving Newton's equations of motion (see Section 2.3).

At a generic time step n, the total force acting on the particle is given by

$$\mathbf{F}_p^* = M_f \frac{d\mathbf{u}_p^n}{dt} - \sum_b \mathbf{F}_b^n \Delta V_p + (M_p - M_f)\mathbf{g}. \tag{2.18}$$

Instead of using $\mathbf{F}_p^*$ from the current time step, we make use of the information from the previous W time steps in order to filter out spurious oscillations. In a general form, we solve Newton's equation of motion, using the force term defined as

$$\mathbf{F}_p^n = \Phi(N,n,\mathbf{F}_p^*,\mathbf{F}_p^{n-1},\mathbf{F}_p^{n-2},\cdots,\mathbf{F}_p^{n-W}), \tag{2.19}$$

where Φ represents a model function that is used to smooth the data from the $W+1$ data points and N represents the order of the model function.

In what follows, we consider two possible implementations of Φ:

1. The new value of the forcing term is calculated as the moving average of the previous W time steps:
$$\mathbf{F}_p^n = \frac{\mathbf{F}_p^* + \mathbf{F}_p^{n-1} + \mathbf{F}_p^{n-2} + \cdots + \mathbf{F}_p^{n-W}}{W+1}. \tag{2.20}$$

2. The new value of the forcing term is calculated applying a order N polynomial regression on the previous:
$$\mathbf{F}_p^n = \sum_{i=0}^{N} a_i n^i, \tag{2.21}$$
where a_i are the regression parameters.

In what follows, we will label IBM+MA the method making use of the moving average in the calculation of the forcing term. We will also refer to IBM+LF and IMB+QF to indicate the method employing respectively a linear fit or a quadratic fit in the calculation of the forcing term. Several other kernels can be considered for smoothing the forcing term, such as splines, Savitzky-Golay, and many more (see e.g. Ref. [23]), however our aim here is that even very simple methods are sufficient to significantly improve the stability of IBM.

In Fig. 2 we provide an example (see Section 2.4 for the definition of the setup), showing how employing IBM+MA (red lines), with $W=4$, it is possible to stabilize the time dynamic of the system with respect to a basic IBM implementation (blue lines).

3 Results and discussion

In this section, we evaluate the force stabilization method proposed in the previous section, by performing simulations in the context of sedimentation/rising of particles subject to gravity in an enclosed domain.

We compare the results obtained with experimental and numerical results available in the literature, for both heavy and light particles. In order to compare with previous studies in what follows we will focus on spherical particles; however, we shall remark that the methodology here presented is general and valid in principle for particles of any shape.

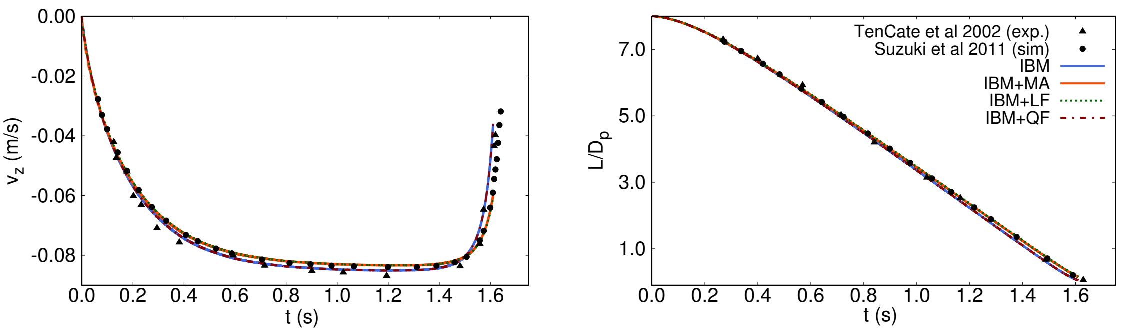
3.1 Test problem A

We start by considering the sedimentation of a heavy particle under gravity in an enclosed domain, working in a parameter range for which the standard IBM implementation provides accurate results.

We use the same setup used in experiments by TenCate et al. [24]. The test case involves an enclosed domain of size 10 cm$\times$10 cm$\times$16 cm filled with a liquid of density $\rho_f = 962$ kg/m^3 and dynamic viscosity $\mu_f = 0.113$ Pa$\cdot$ s. A solid particle of diameter $D_p = 1.5$ cm and density $\rho_p = 1120$ kg/m^3 is released from a position such that the lower surface of the particle is 12 cm from the bottom of the vessel under gravity ($g = 9.8$ m/s^2). These quantities are defined in numerical units on a grid of size $200\times200\times320$, with the particle defined by 3156 Lagrangian markers. The relaxation time is set to $\tau = 0.6$ and the gravity in LU is $g = -5.346\times10^{-4}$. For the current setup, no-slip boundary conditions are applied to both top/bottom and side walls.

In Fig. 3 we show the profiles for the vertical components of velocity and position (normalised with particle diameter D_p) of the particle as a function of time. Our numerical results compare well with both the experimental data of TenCate et al. [24], as well as with the numerical results of Suzuki et al. [2].

The force stabilization is applied using data from $W = 5$ time steps, for both



(a) vertical velocity of particle over time

(b) vertical position of particle, normalized over its diameter, over time

Figure 3: Time evolution of (a) the vertical velocity and (b) the vertical position of a heavy ($\rho_p/\rho_f = 1.1642$) particle (normalized over its diameter D_p), undergoing sedimentation under gravity. The plots show a comparison of the IBM (blue line) with the experimental results of TenCate et al. [24], as well as with the numerical results of Suzuki et al. [2]. The red curves show the results obtained applying a moving average in the calculation of the force (IBM+MA), the green curves employ a linear fit (IBM+QF), whereas in dark red we show the results obtained with a quadratic fit (IBM+QF). In all these cases we use the same window length to perform the smoothed calculation of the force ($W = 5$ LU).

IBM+MA/LF/QF. We observe that the moving average and a linear fit produce results slightly different from those of the basic IBM implementation, though still in good agreement with experimental data. The quadratic fit instead perfectly matches the results of the basic IBM implementation.

The value of W should be carefully chosen: if taken to be too large in comparison to the transients of the physical problem, then the low-order methods will not be able to capture the correct dynamics. Moreover, a too large value for W would also lead to large numerical dissipation for the kinetic energy, which can significantly impact the particle dynamics leading to a non-physical time evolution of the system. In Fig. 4 we show the results for v_z, obtained using $W = 9$, zooming in a time window of 0.05 s.

The results using IBM+MA slightly deviate from the correct dynamics. IBM+LF, on the other hand, qualitatively follows the basic IBM implementation but introduces oscillations. The oscillations are removed when adopting IMB+QF.

3.2 Test problem B

We now consider a second test case, to highlight the benefits of our force stabilization method. We replicate the same setup previously used by several authors (see e.g. [25–27]), in which a spherical particle of diameter D_p is released from the center of a vertical channel of width L. Under gravity the particle either falls or rises, reaching a terminal velocity. The particle Reynolds number chosen for the problem is such that the solution falls within the Stokes flow regime. However, since the walls are placed at a finite distance from the particle, the final velocity of the particle will be less than

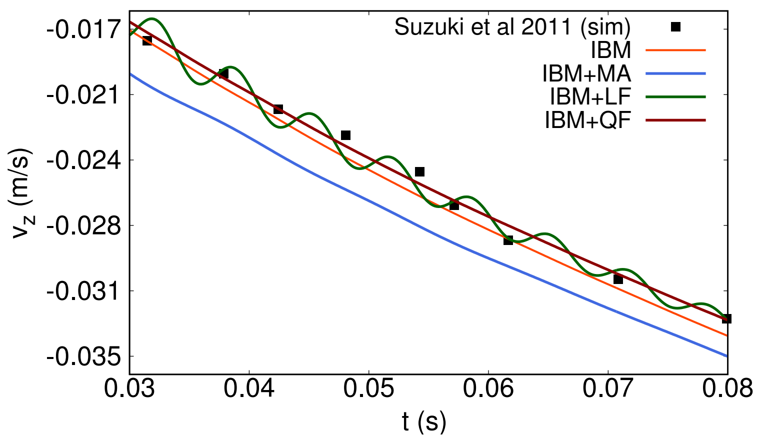


Figure 4: Vertical velocity (v_z) of a sedimenting spherical particle over time. The particle chosen here is heavy with a particle to fluid density ratio of ($\rho_p/\rho_f\!=\!1.1642$) A window length of $W\!=\!9$, which corresponds to 0.0015 sec in physical units, has been used for all the smoothing techniques shown in the plot. The data used for comparison has been obtained from Suzuki et al. [2].

the Stokes velocity. It was shown experimentally by Miyamura [25], that it is possible to define a relation between the final velocity of the particle, normalized with the unbounded Stokes velocity (u_p/u_0), and the ratio between the particle diameter and the channel width (D_p/L). Similar studies were performed by Gupta et al. [27] and Aidun et al. [26] numerically.

We consider both light and heavy particles, simulated on a domain consisting of $L\times L\times 4L$ nodes, with $L\!=\!32$ (coarse) and $L\!=\!64$ (fine). We work in the Stokes regime ($\mathrm{Re}_\mathrm{p}\ll 1$) with Reynolds number varied within the range [0.0006,0.02222] for heavy particles, and [0.00012,0.00444], for light particles. A set of particle sizes were simulated for different D_p/L values ranging from 0.1875 ($D_p\!=\!6$ for coarse and $D_p\!=\!12$ for fine) to 0.625 ($D_p\!=\!20$ for coarse and $D_p\!=\!40$ for fine). A zero-velocity inlet is placed at the top boundary, while a convective outlet is used at the bottom boundary, with no-slip boundary conditions at the side walls. In Fig. 5 we show the result for obtained employing the IBM with the moving average technique (IBM+MA), for both a heavy ($\rho_p/\rho_f\!=\!2.0$) and a light particle ($\rho_p/\rho_f\!=\!0.8$). The results show good agreement with [25] and [27].

We have also included results using a higher grid resolution of size $L\times L\times 4L$, with $L=64$, to highlight the fact that the discrepancies observed for low values of the D_p/L parameter are due to the particle not being resolved with enough accuracy.

We conclude our analysis by evaluating the numerical stability and accuracy when varying the particle-to-fluid density ratio (ρ_p/ρ_f). We consider two examples, fixing the ratio of the particle diameter to channel width (D_p/L) respectively to 0.25 and 0.5. The particle diameter is set to $D_p=16$ LU that gives us two domains $32\times 32\times 256$ and $64\times 64\times 256$ for each D_p/L. To describe the boundary of the particle 984 Lagrangian markers were used. The particle Reynolds number (Re_p) is set to 0.001.

In Fig. 6 we show that while the basic IB method is stable up to density ratio of 0.4,

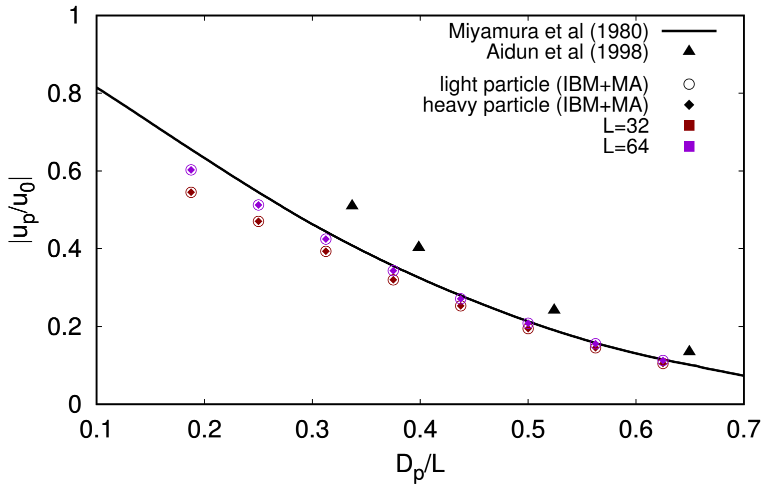
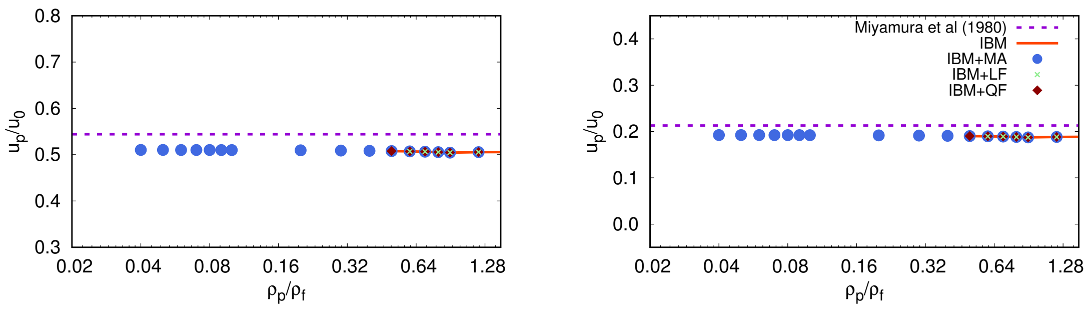


Figure 5: Sedimentation and rising of a spherical particle in an enclosed channel. The plot shows the terminal velocities (u_p) normalized over the Stokes settling velocity (u_0), versus the ratio between the particle diameter and the channel width (D_p/L). We consider both light ($\rho_p/\rho_f = 0.8$) and heavy ($\rho_p/\rho_f = 2$) particles, and compare against the results reported by Miyamura et al. [25] and Abhineet et al. [27]. Note that due to the normalization the results presented do not depend on the ratio between the particle density and the fluid density.



(a) $D_p/L = 0.25$ (b) $D_p/L = 0.5$

Figure 6: Rising of a spherical particle under gravity in an enclosed domain. The plot shows the terminal velocities (u_p) normalised with the Stokes settling velocity (u_0) for a range of particle-to-fluid density ratios (ρ_p/ρ_f). Two sets of particle diameter to the channel width ratios (D_p/L) has been shown here. A window length of $W = 5$ has been used for all the smoothing techniques used here. The reference data is taken from Miyamura et al. [25].

by employing the moving average in the calculation of the forcing term (IBM+MA) it is possible to obtain stable numerical results for densities ratios as low as 0.04.

Moreover, the accuracy of the solution is independent of both the density ratio and the chosen force stabilizing method, and well compares with the reference data from Miyamura et al. [25].

4 Conclusion

In this work we have implemented a simple, effective and computationally efficient technique for the simulation of light particles in complex fluid flows. Our method allows simulations with a particle to fluid density ratio as low as 0.04, which significantly improves over Uhlmann's [10] IBM implementation. It is important to note that as we have observed the instability is also dependent on the Reynolds number of the system and can show higher level of stability at very low Re_p (Stokes regime). This will be an object of further analysis in the future.

We avoid the explicit calculation of the rigid body approximation by calculating the total force acting on the particle using information from data from previous iterations. This allows reducing, or even completely removing, the oscillations observed in the basic IBM implementation. Moreover, the approach is computationally efficient since the overheads introduced by the smoothing of the forcing term are negligible: to give an example, the execution time of the IBM combined with a moving average of the forcing term with $W=4$ is increased of just about $\approx 2\%$ in comparison to the basic IBM implementation.

We have validated our implementation using two test cases considering both heavy and light particles. Our results are in good agreement with both experimental and numerical data from previous studies.

In an extended version of the present work, we will further investigate the role of the time window size, used to apply the force stabilization method, in order to define an optimal value for this parameter from the target physical parameters of the problem. Moreover, we will provide a quantitative performance comparison with numerical methods employing the rigid body approximation.

Acknowledgments

This publication is part of the project Fundamental of Inkjet Printing (FIP) (with project number FIP NWOI-18.0166) in collaboration with Canon Production Printing Netherlands B.V and the work is (partly) sponsored by NWO domain Science for the use of supercomputer facilities. We would like to thank Dr. Chong Shen Ng for the useful discussions.

Appendices

A Calculation of the angular rotation of the particle

In this appendix section, we introduce the procedure adopted for updating the orientation of the particle over time. The method relies on quaternions and it is general and applicable to particles of any shape.

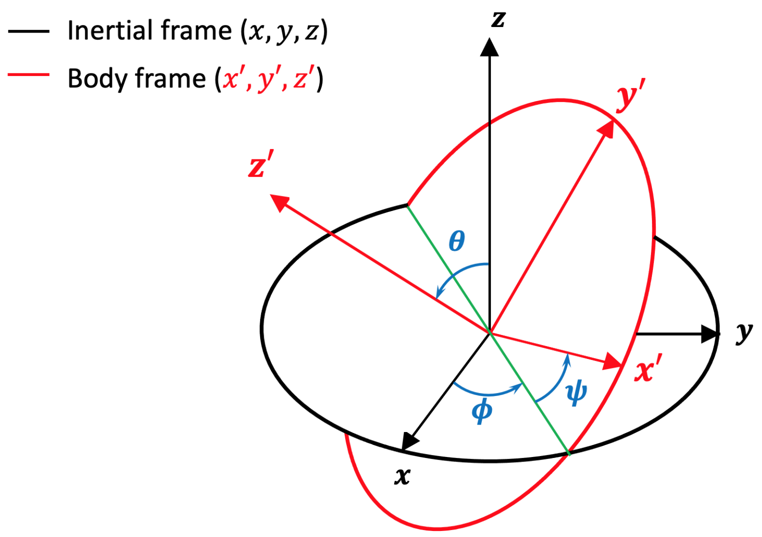


Figure 7: Euler angles θ,ϕ,ψ. The $\mathbf{x}=<x,y,z>$ axis, shown by the black lines and ellipse denotes the inertial frame of reference which is attached to the Cartesian grid (Eulerian nodes). The $\mathbf{x}'=<x',y',z'>$ axis refers to the body frame of reference attached to the particle and it is shown by the red lines and ellipse.

We start by defining the inertial frame $\mathbf{x}=<x,y,z>$ of the Eulerian fluid nodes, and the body frame of reference, which we label as $\mathbf{x}'=<x',y',z'>$, which is attached to the center of mass of the particle.

The moment of inertia for the particle is known in the body frame of reference; in what follows we will denote it as I'_{xx},I'_{yy},I'_{zz}.

In order to transform physical quantities from the inertial frame to the body frame we introduce the quaternion variables:

$$\begin{aligned} q_0&=\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi+\psi}{2}\right),\\ q_1&=\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\phi-\psi}{2}\right),\\ q_2&=\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\phi-\psi}{2}\right),\\ q_3&=\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\phi+\psi}{2}\right), \end{aligned} \tag{A.1}$$

where θ,ϕ,ψ are the Euler angles.

The transformation matrix is defined as:

$$A=\begin{pmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_0q_1+q_2q_3) & 2(q_1q_3-q_0q_2)\\ 2(q_1q_2-q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3+q_0q_1)\\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{pmatrix}. \tag{A.2}$$

It is possible to calculate the angular momentum in the inertial frame using Eq. (2.13), to then transform it from the inertial frame to body frame of reference via

$$(I'\omega') = A(I\omega), \tag{A.3}$$

thus allowing the calculation of the angular velocity in body frame of reference (ω').

Finally, one can calculate the rate of change of orientation of the particle in body frame of reference using the angular velocity (ω') by introducing a quantity that represents the rate of change of quaternions $\dot{\mathbf{q}}$:

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{q_0} \\ \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} 0 \\ \omega'_x \\ \omega'_y \\ \omega'_z \end{bmatrix}. \tag{A.4}$$

Using the above equation it is possible to update the values of the quaternions (q_0,q_1,q_2,q_3), which can be then used to update the orientation of the particle in the next time step.

Note that at each time step it is also necessary to normalize the quaternions in order to ensure that $q_0^2+q_1^2+q_2^2+q_3^2=1$.

B Lagrangian markers distribution on particle

In this Appendix section we briefly summarize the algorithm adopted for uniformly distributing the Lagrangian markers on the surface of a particle, allowing the description of the particle in the IBM.

Following Ref. [28] we perform the following steps:

1. Orient the spheroid so that the major axis aligns with the vertical axis.
2. Choose the Lagrangian grid spacing (Δs) to be between 0.7 to 0.9 times the Eulerian grid spacing (Δh).
3. Divide the spheroid into a number of strips along the major axis with a width equal to the Δs. Each strip should be circular in cross-section.
4. Along each strip distribute the Lagrangian markers with a spacing equal to Δs.

The number of Lagrangian markers along each strip will depend on the circumferential length of the strip. This will ensure that the strips nears the poles will have less number of Lagrangian markers than the strips near the equator and maintain uniformity.

The Lagrangian grid spacing (Δs) is chosen to be some degree less than the Eulerian node spacing (Δh). This is done so as to prevent leakage of fluid through the particle surface and hence satisfy no-penetration boundary condition.

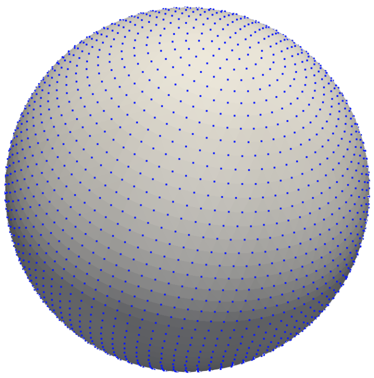


Figure 8: Lagrangian marker distribution on a spherical particle. 3156 Lagrangian markers is used to generate the spherical particle of diameter 30 LU. The Lagrangian spacing is 0.9 times of the Eulerian node spacing ($\Delta s \approx 0.9\Delta h$).

Also, decreasing the Lagrangian grid spacing too much will increase the number of Lagrangian markers and hence increase computational cost. Ideally the Lagrangian grid spacing (Δs) should be around 0.7 to 0.9 times the Eulerian grid spacing (Δh).

Fig. 8 illustrates an example of a spherical IB particle with 3156 Lagrangian markers distributed on the surface which is used in test problem A in Section 3.1. The spherical particle is of diameter 30 LU with $\Delta s \approx 0.9\Delta h$

C List of parameters and values for the test cases

In this appendix section we summarize all the relevant physical parameters needed for reproducing the numerical results presented in Section 3. In Table 1 we provide the pa-

Table 1: List of physical parameters used in Test problem A (Section 3.1).

	Physical Units	Lattice Units
$L_x \times L_y \times L_z$	$10\times 10\times 16$ [cm]	$200\times 200\times 320$
D_p	1.5 [cm]	15
$\boldsymbol{x}_p(t=0)$	(5,5,12.75) [cm]	(100.5,100.5,255.5)
ρ_f	962 [kg m^{-3}]	1
ρ_p	1120 [kg m^{-3}]	1.1642
μ_f	0.113 [Pa s]	
ν_f	1.1746×10^{-4} [kg m^{-2}]	0.0776
τ		0.7328
g	9.8 [m s^{-2}]	5.346×10^{-4}

Table 2: List of physical parameters used in Test problem B (Section 3.2).

	Heavy particle		Light particle	
	Coarse grid	Fine grid	Coarse grid	Fine grid
$L_x \times L_y \times L_z$	$32\times32\times128$	$64\times64\times256$	$32\times32\times128$	$64\times64\times256$
$x_p(t=0)$	16,16,64	32,32,128	16,16,64	32,32,128
ρ_f	1.0	1.0	1.0	1.0
ρ_p	2.0	2.0	0.8	0.8
ν_f	0.1	0.1	0.1	0.1
τ	0.8	0.8	0.8	0.8
g	5×10^{-7}	5×10^{-8}	5×10^{-7}	5×10^{-8}

rameters used for test problem A (Section 3.1), while in Table 2 we provide the parameters for reproducing test problem B (Section 3.2).

References

[1] T. Bodnar, G. Galdi, S. Necasova, Particles in Flows, Springer, 2017. doi:10.1007/978-3-319-60282-0.

[2] K. Suzuki, T. Inamuro, Effect of internal mass in the simulation of a moving body by the immersed boundary method, Computers & Fluids 49 (1) (2011) 173–187. doi:10.1016/j.compfluid.2011.05.011.

[3] C. Veldhuis, A. Biesheuvel, L. van Wijngaarden, D. Lohse, Motion and wake structure of spherical particles, Nonlinearity 18 (1) (2004) C1–C8. doi:10.1088/0951-7715/18/1/000.

[4] C. Veldhuis, A. Biesheuvel, An experimental study of the regimes of motion of spheres falling or ascending freely in a newtonian fluid, International Journal of Multiphase Flow 33 (10) (2007) 1074–1087. doi:10.1016/j.ijmultiphaseflow.2007.05.002.

[5] C. Veldhuis, A. Biesheuvel, D. Lohse, Freely rising light solid spheres, International Journal of Multiphase Flow 35 (4) (2009) 312–322. doi:10.1016/j.ijmultiphaseflow.2009.01.005.

[6] C. S. Peskin, Flow patterns around heart valves: A numerical method, Journal of Computational Physics 10 (2) (1972) 252–271. doi:10.1016/0021-9991(72)90065-4.

[7] C. S. Peskin, Numerical analysis of blood flow in the heart, Journal of Computational Physics 25 (3) (1977) 220–252. doi:10.1016/0021-9991(77)90100-0.

[8] R. Mittal, G. Iaccarino, Immersed boundary methods, Annual Review of Fluid Mechanics 37 (1) (2005) 239–261. doi:10.1146/annurev.fluid.37.061903.175743.

[9] G. Iaccarino, R. Verzicco, Immersed boundary technique for turbulent flow simulations , Applied Mechanics Reviews 56 (3) (2003) 331–347. doi:10.1115/1.1563627.

[10] M. Uhlmann, An immersed boundary method with direct forcing for the simulation of particulate flows, Journal of Computational Physics 209 (2) (2005) 448–476. doi:10.1016/j.jcp.2005.03.017.

[11] Z.-G. Feng, E. E. Michaelides, The immersed boundary-lattice boltzmann method for solving fluid–particles interaction problems, Journal of Computational Physics 195 (2) (2004) 602–628. doi:10.1016/j.jcp.2003.10.013.

[12] T. Kempe, J. Fröhlich, An improved immersed boundary method with direct forcing for the simulation of particle laden flows, Journal of Computational Physics 231 (9) (2012) 3663–3684. doi:10.1016/j.jcp.2012.01.021.

[13] S. Schwarz, T. Kempe, J. Fröhlich, A temporal discretization scheme to compute the motion of light particles in viscous flows by an immersed boundary method, Journal of Computational Physics 281 (2015) 591–613. doi:10.1016/j.jcp.2014.10.039.

[14] W.-P. Breugem, A second-order accurate immersed boundary method for fully resolved simulations of particle-laden flows, Journal of Computational Physics 231 (13) (2012) 4469–4498. doi:10.1016/j.jcp.2012.02.026.

[15] A. Spizzichino, S. Goldring, Y. Feldman, The immersed boundary method: application to two-phase immiscible flows, Commun. Comput. Phys 25 (1) (2019) 107–134. doi:10.4208/cicp.OA-2018-0018.

[16] Y. Li, E. Jung, W. Lee, H. G. Lee, J. Kim, Volume preserving immersed boundary methods for two-phase fluid flows, International Journal for Numerical Methods in Fluids 69 (4) (2012) 842–858. doi:10.1002/fld.2616.

[17] S. Succi, The Lattice Boltzmann Equation: For Complex States of Flowing Matter, OUP Oxford, 2018. doi:10.1093/oso/9780199592357.001.0001.

[18] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, E. M. Viggen, The Lattice Boltzmann Method - Principles and Practice, Springer, 2016. doi:10.1007/978-3-319-44649-3.

[19] X. Shan, The mathematical structure of the lattices of the lattice Boltzmann method, Journal of Computational Science 17 (2016) 475 – 481. doi:10.1016/j.jocs.2016.03.002.

[20] P. Bhatnagar, E. Gross, M. Krook, A model for collision processes in gases. i. small amplitude processes in charged and neutral one-component systems, Phys. Rev. 94 (1954) 511–525. doi:10.1103/PhysRev.94.511.

[21] Z. Guo, C. Zheng, B. Shi, Discrete lattice effects on the forcing term in the lattice boltzmann method, Phys. Rev. E 65 (2002) 046308. doi:10.1103/PhysRevE.65.046308.

[22] H. Goldstein, C. Poole, J. Safko, Classical Mechanics, 3rd ed., Vol. 70, 2002. doi:10.1119/1.1484149.

[23] Accurate computation of surface stresses and forces with immersed boundary methods, Journal of Computational Physics 321 (2016) 860–873. doi:10.1016/j.jcp.2016.06.014.

[24] A. ten Cate, C. H. Nieuwstad, J. J. Derksen, H. E. A. Van den Akker, Particle imaging velocimetry experiments and lattice-boltzmann simulations on a single sphere settling under gravity, Physics of Fluids 14 (11) (2002) 4012–4025. doi:10.1063/1.1512918.

[25] A. Miyamura, S. Iwasaki, T. Ishii, Experimental wall correction factors of single solid spheres in triangular and square cylinders, and parallel plates, International Journal of Multiphase Flow 7 (1) (1981) 41–46. doi:10.1016/0301-9322(81)90013-6.

[26] A. C. K., L. YANNAN, E.-J. DING, Direct analysis of particulate suspensions with inertia using the discrete boltzmann equation, Journal of Fluid Mechanics 373 (1998) 287–311. doi:10.1017/S0022112098002493.

[27] A. Gupta, H. J. Clercx, F. Toschi, Simulation of finite-size particles in turbulent flows using the lattice boltzmann method, Communications in Computational Physics 23 (3) (2018) 665–684. doi:10.4208/cicp.OA-2016-0268.

[28] A. Eshghinejadfard, A. Abdelsamie, G. Janiga, D. Thévenin, Direct-forcing immersed boundary lattice boltzmann simulation of particle/fluid interactions for spherical and non-spherical particles, Particuology 25 (2016) 93–103. doi:10.1016/j.partic.2015.05.004.