Introduction to the immersed boundary method

by Timm Krüger (info@timm-krueger.de), last updated on September 27, 2011

1 Motivation

1.1 Hydrodynamics and boundary conditions

The incompressible Navier-Stokes equations,

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

are partial differential equations. Hence, initial and boundary conditions are required to find solutions.
Usually, boundary conditions in the lattice Boltzmann method (LBM) are introduced on the population level, i.e., the populations f_i are locally manipulated in such a way that the moments (e.g., pressure, velocity) give the desired results. Typical boundary conditions of this kind are bounce-back or regularized boundary conditions.
There is another possibility: The forcing term $\boldsymbol{f}$ in eq. (1) may be used to mimic a boundary condition. A local force density is introduced to modify the flow field. When applied correctly, this can be used to realize deformable or rigid objects in the flow. The advantage of this approach is that complex geometries can be easily implemented.
For this reason, it is worthwhile to separate the forcing term into two parts, $\boldsymbol{g}+\boldsymbol{f}$, where $\boldsymbol{g}$ contains all physical contributions (such as gravity) and $\boldsymbol{f}$ contains the force field due to the boundary condition. In the following, $\boldsymbol{g}$ is arbitrary and not discussed further. Note that this approach is similar to that in the Shan-Chen model [1].
Eventually, for a given problem, it is necessary to

1. define the macroscopic boundary conditions and to
2. implement them in the LBM in a consistent way.

1.2 Complexity of boundary conditions

Boundaries in practical applications are usually complex because they may be

- arbitrarily shaped (curved),
- moving in space,
- deformable.

This complexity is difficult to handle:

- The resulting equations are often too complex to be solved analytically.
- If the boundaries are deformable, additional constitutive laws are required.
- How shall the presence of the boundary be translated into the LBM language?

1.3 Staircase approximation

The simplest way to introduce non-planar boundaries in LBM simulations is via a staircase approximation of the boundary and the half-way bounce-back scheme, cf. fig. 1.
Advantages:

- Everything lives on the regular grid.

Figure 1: Bounce-back (left) is often applied for non-aligned walls. In this case, the boundary is approximated by a staircase (right).

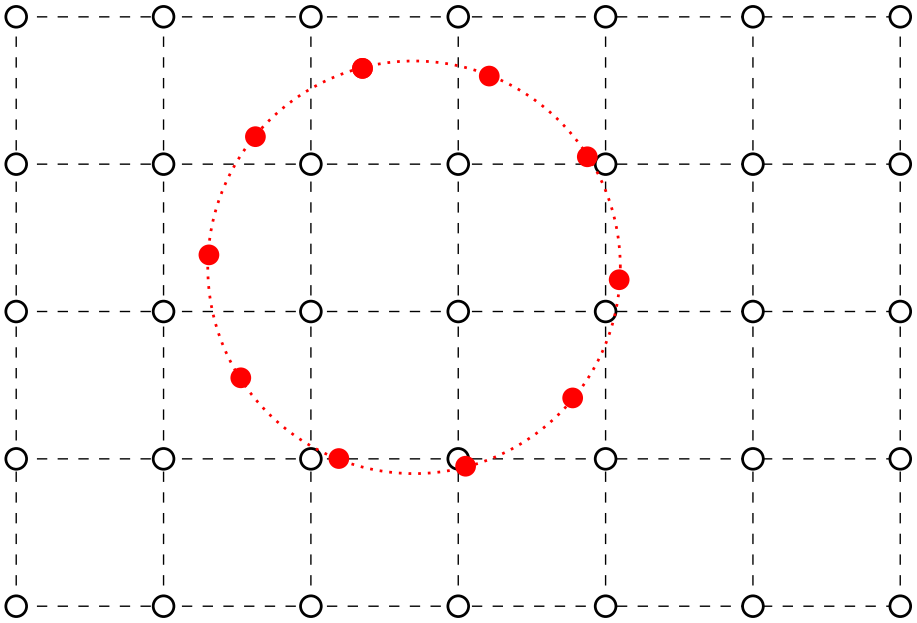

Figure 2: Cylinder with conform mesh arbitrarily positioned in the regular fluid domain.

- Standard bounce-back (which is well-known) is used.
- Basic objects (spheres, cylinders) are simple to convert to a staircase.

Disadvantages:

- What happens if boundaries move? Fluid nodes have to be destroyed and created.
- What happens if boundaries are deformable? Constitutive laws have to be combined with bounce-back, which can be tricky.
- Staircase bounce-back can be inaccurate if the LBM relaxation time τ is not well chosen.

1.4 Non-conform meshes

Another way to introduce complex boundaries is by defining a new, non-conform mesh which is fitting the boundary. This new mesh is generally allowed to move and deform, cf. fig. 2.
Advantages:

- direct description of the boundary (simplifying discription of constitutive laws)
- easier handling of deformation and motion

New question:

- how to couple to the fluid grid?

A bi-directional coupling is crucial:

1. The fluid has to know about the presence of the boundary.

2. The boundary, if moving and deformable, has to know about the fluid stress on its surface.

One possible approach to tackle the problem is the immersed boundary method (IBM) which will be described in the following. Another way is the interpolated bounce-back [2, 3] which is not covered here.

2 The immersed boundary method

2.1 Short historical overview

The IBM has been introduced by Peskin in the 1970s [4–6]. It has originally been used to simulate flow patterns in the heart.
The first simulation of a deformable red blood cell with the IBM has been performed by Eggleton and Popel [7] in 1998.
Feng and Michaelides [8] were the first to combine the LBM and the IBM for the simulation of rigid disks in 2D in 2004.
Red blood cell simulations with the LBM and IBM in 2D and 3D have been performed from 2007 on [9–11].
An implicit IBM combined with the LBM for rigid objects has been introduced in 2009 [12].
It is important to stress that the IBM has been introduced far before the LBM has been known. Both are completely independent concepts, but they can be combined.

2.2 Mathematical basis

In the IBM, there are two coordinate systems, cf. fig. 2 and fig. 3:

1. Eulerian grid for the fluid: stationary and regular

2. Lagrangian mesh for the boundaries: generally non-stationary and unstructured (i.e., no intrinsic coordinate system)

Information between both coordinate systems is communicated via interpolations.
The bi-directional coupling bases on two principles:

1. no-slip condition for the velocity at the boundary (special case of a Dirichlet boundary condition for the velocity)

2. Newton's law: 'actio = reactio'

The fluid is aware of the boundary only through a forcing term. There is no direct boundary condition for the fluid (especially not for the populations). This means that the populations can cross the boundaries without seeing them, but the macroscopic fluid behaves as if there was a boundary. This is one of the central ideas of the combined IB-LBM algorithm. Additionally, the fluid fills the entire space, even inside the boundary region. This significantly simplifies things but can also lead to additional difficulties (e.g., motion of internal fluid at higher Reynolds numbers [13]).

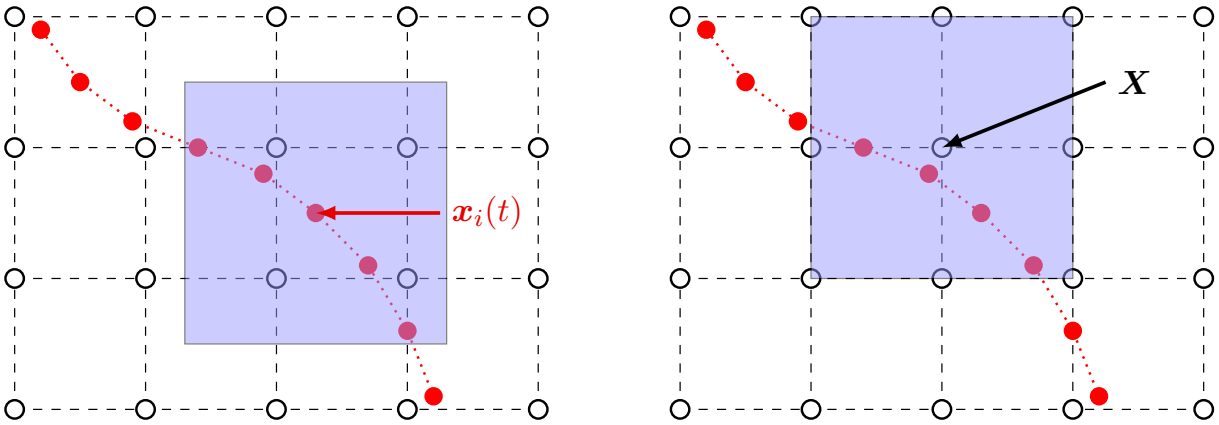

Figure 3: Illustration of velocity interpolation (left) and force spreading (right). The velocity of node i is interpolated from the lattice nodes within the square region. The force acting at lattice node $\boldsymbol{X}$ is the combination of all contributions within the square region.

Table 1: Notations used throughout the lecture. All quantities are functions of time, except for the lattice grid coordinates which are fixed by definition.

$\boldsymbol{X}$	fixed lattice grid coordinates (fluid)
$\boldsymbol{x}_i(t)$	position of boundary node i
$\boldsymbol{u}(\boldsymbol{X},t)$	fluid velocity
$\dot{\boldsymbol{x}}_i(t)$	velocity of boundary node i
$\boldsymbol{f}(\boldsymbol{X},t)$	fluid force density (force per volume)
$\boldsymbol{F}_i(t)$	force acting on node i

2.3 Interpolation equations

The complete set of equations for the IBM are given by Peskin [6]. In the following, a simplified case is discussed:

- The boundary is 2D immersed in 3D space, i.e., it is a mere surface.
- The fluid in the entire volume is identical (density, viscosity).
- The boundary is massless.

The notations used throughout this lecture are given in tab. 1. Based on these and setting the lattice constant to unity ($\Delta x = 1$), the discretized IBM equations read

$$\dot{\boldsymbol{x}}_i(t+1) = \sum_{\boldsymbol{X}} \boldsymbol{u}(\boldsymbol{X}, t+1)\Delta(\boldsymbol{x}_i(t) - \boldsymbol{X}) \tag{2}$$

and

$$\boldsymbol{f}(\boldsymbol{X}, t) = \sum_{i} \boldsymbol{F}_i(t)\Delta(\boldsymbol{x}_i(t) - \boldsymbol{X}). \tag{3}$$

The function $\Delta(\boldsymbol{x}_i(t) - \boldsymbol{X})$ is a discretized version of the Dirac delta distribution. It will be discussed below.
The interpretation of eq. (2) is that the boundary velocity matches the ambient fluid velocity (no-slip condition). Ideally, it would be written in the form

$$\dot{\boldsymbol{x}}_i(t+1) = \boldsymbol{u}(\boldsymbol{x}_i(t), t+1), \tag{4}$$

but the fluid velocity is only known at grid nodes $\boldsymbol{X}$ and has to be interpolated to arbitrary points $\boldsymbol{x}_i$.

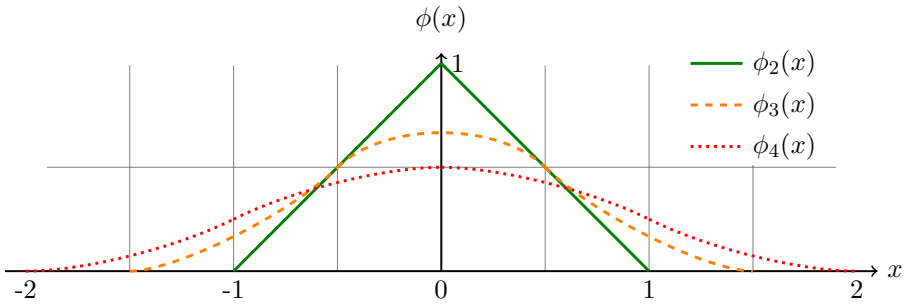

Figure 4: IBM interpolation stencils ϕ_2, ϕ_3, and ϕ_4.

Eq. (3) reflects Newton's law 'actio = reactio'. The basic idea is that all forces which act on the boundary also have to act on the ambient fluid in order to ensure total momentum conservation. A sketch of the interpolation and spreading operations is shown in fig. 3. The numerical algorithm of the coupled immersed boundary lattice Boltzmann method (IB-LBM) will be described below.
Note that the IBM is not responsible for providing a suitable force $\boldsymbol{f}$. It depends on the physics of the systems (e.g., rigid or deformable particles/walls) and the numerical implementation (e.g., explicit, implicit) how the forcing term is eventually obtained. This is the reason for the large variety of articles dealing with this problem.

2.4 Interpolation stencils

Eqs. (2) and (3) contain an interpolation function (also called interpolation kernel) $\Delta(\boldsymbol{r})$ with $\boldsymbol{r} = \boldsymbol{x}_i(t) - \boldsymbol{X}$. It is not directly obvious which function to take in order to approximate Dirac's delta distribution. The procedure to find suitable functions is described by Peskin [6]. Here, only the basic ideas and the results are given.
The fundamental claims are:

- The interpolations should be short-ranged with a finite cut-off length. This is required to reduce computational overhead.
- Momentum and angular momentum have to be identical when evaluated either in the Eulerian or the Lagrangian frame.
- The lattice structure of the Eulerian mesh should be hidden as much as possible.

It is convenient to factorize the interpolation function, $\Delta(\boldsymbol{r}) = \phi(x)\phi(y)\phi(z)$, i.e., each lattice direction contributes independently. This is not essential but simplifies the procedure. Peskin derived a series of interpolation stencils which are also shown in fig. 4:

$$\phi_2(x) = \begin{cases} 1 - |x| & \text{for} \quad 0 \le |x| \le 1 \\ 0 & \text{for} \quad 1 \le |x| \end{cases}, \tag{5}$$

$$\phi_3(x) = \begin{cases} \frac{1}{3}(1 + \sqrt{1 - 3x^2}) & \text{for} \quad 0 \le |x| \le \frac{1}{2} \\ \frac{1}{6}(5 - 3|x| - \sqrt{-2 + 6|x| - 3x^2}) & \text{for} \quad \frac{1}{2} \le |x| \le \frac{3}{2} \\ 0 & \text{for} \quad \frac{3}{2} \le |x| \end{cases}, \tag{6}$$

$$\phi_4(x) = \begin{cases} \frac{1}{8}\left(3 - 2|x| + \sqrt{1 + 4|x| - 4x^2}\right) & \text{for} \quad 0 \le |x| \le 1 \\ \frac{1}{8}\left(5 - 2|x| - \sqrt{-7 + 12|x| - 4x^2}\right) & \text{for} \quad 1 \le |x| \le 2 \\ 0 & \text{for} \quad 2 \le |x| \end{cases}. \tag{7}$$

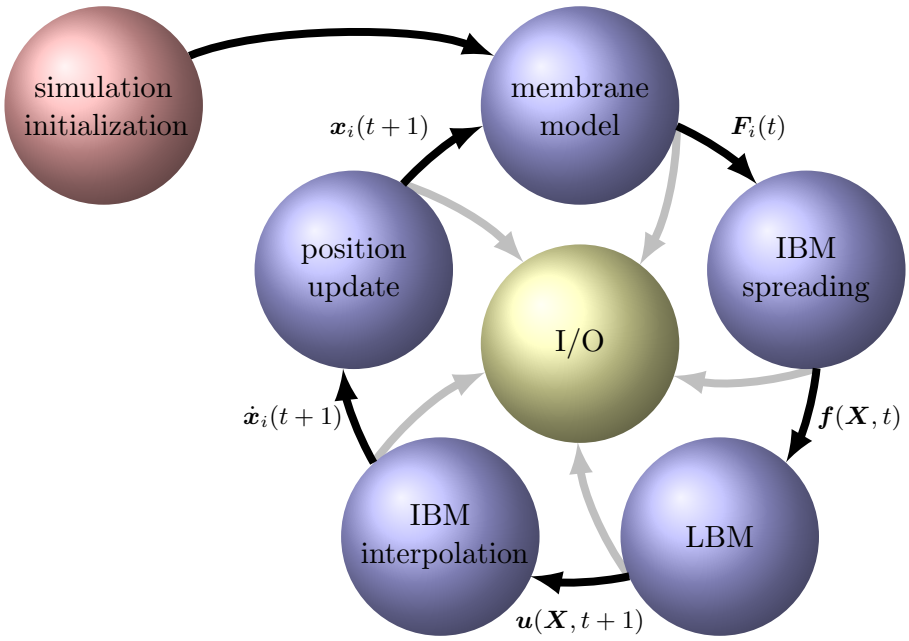

Figure 5: Overview of the simulation algorithm for deformable particles. Data may be written to the disk when required.

ϕ_4 fulfills all of Peskin's requirements, but it also leads to a diffuse boundary since the interpolation range is rather large. ϕ_3 also fulfills all of Peskin's requirements, but it cannot be employed in central finite difference schemes. However, this is not a problem here because LBM is used instead. ϕ_2 is most efficient in terms of computing time and leads to the sharpest boundaries, but the lattice structure is not well hidden, i.e., the resulting flow field can be less smooth. Sometimes, $\phi_4(x)$ is replaces by another stencil which has nearly the same shape,

$$\phi_4^c(x) = \begin{cases} \frac{1}{4}(1+\cos(\frac{\pi x}{2})) & \text{for} \quad 0 \leq |x| \leq 2 \\ 0 & \text{for} \quad 2 \leq |x| \end{cases}. \tag{8}$$

Up to this point, all relevant concepts and ideas behind the IBM have been summarized. The remainder of this document deals with the question how to obtain suitable expressions for the forcing term $\boldsymbol{f}$.

3 IBM for deformable particles

The simulation of deformable particles is important, e.g., for blood flow, food industry, cosmetics etc. Additionally to the correct description of the suspending fluid via LBM, a constitutive law for the behavior of the particles has to be defined. This constitutive law relates the deformation of the particle and the stresses/forces.

3.1 Algorithm

The simulation algorithm for deformable membranes can be summarized like this, cf. also [14] and fig. 5:

1. Compute the membrane forces $\boldsymbol{F}_i(t)$ based on the membrane deformation and using the constitutive law.
2. Spread the forces to the lattice via eq. (3) and obtain the lattice force density $\boldsymbol{f}(\boldsymbol{X},t)$.
3. Use the LBM (including the force density) to obtain the new fluid state $\boldsymbol{u}(\boldsymbol{X},t+1)$.

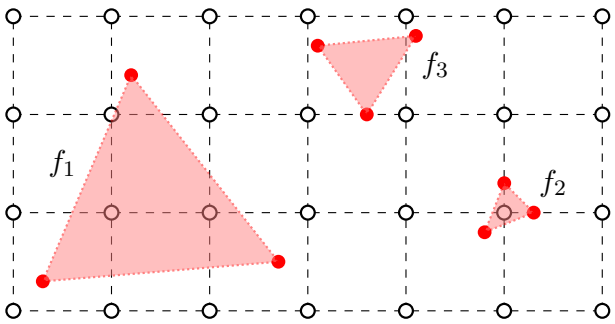

Figure 6: Examples of membrane face elements: f_1 is too coarse, f_2 is too fine, and f_3 is just right.

4. Interpolate the fluid velocity back to the fluid nodes via eq. (2) and obtain $\dot{\boldsymbol{x}}_i(t+1)$.
5. Update all node positions according to $\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \dot{\boldsymbol{x}}_i(t+1)$ (with $\Delta t = 1$).
6. Go back to step 1.

The most demanding part is step 1. A constitutive law has to be specified, and force computation can be quite involved as explained below.

3.2 Membrane energy and forces

For deformable particles, a constitutive law is required. It relates the deformation state to the forces and stresses. The constitutive law contains all the relevant physics of the immersed particles. It is independent of the IBM and has to be provided by the user. There are hyperelastic materials (governed by an elastic energy) or viscoelastic materials (where also viscous dissipation plays a role).
In principle, it is possible to define the forces $\boldsymbol{F}_i$ directly as a function of the configuration state $\{\boldsymbol{x}_i\}$ or even the velocities $\{\dot{\boldsymbol{x}}_i\}$ (not considered here). For example, a simple law may be

$$\boldsymbol{F}_i(t) = k \sum_{j \neq i} \frac{\boldsymbol{x}_j(t) - \boldsymbol{x}_i(t)}{d_{ij}(t)} \frac{d_{ij}(t) - d_{ij}^0}{d_{ij}(t)} \tag{9}$$

with $d_{ij}(t) := |\boldsymbol{x}_i(t) - \boldsymbol{x}_j(t)|$. The sum may run over all neighbors of node i. k is a stiffness coefficient (modulus).
Often, it is not possible to find a simple force law directly. Additionally, momentum or angular momentum conservation may be violated if not done carefully. Instead, a deformation energy density $\epsilon(\{\boldsymbol{x}_i\})$ (energy per area) has to be defined. The force acting on node i can then be recovered by applying the principle of virtual work,

$$\boldsymbol{F}_i = \frac{\partial \epsilon(\{\boldsymbol{x}_j\})}{\partial \boldsymbol{x}_i} A_i, \tag{10}$$

where A_i is the area related to node i (e.g., its Voronoi area). It is usually computationally more efficient to find the derivatives analytically and implement the result directly. Some details can be found, for example, in [14].
Implementing an appropriate constitutive law is a highly problem-specific procedure and not the scope of this lecture.

3.3 Problems and remarks

The success of the algorithm can be either satisfactory or disappointing, depending on the following considerations.
The quality of the membrane mesh plays a major role, cf. fig. 6.

- If the mesh is too coarse, there may be 'holes' in the membrane and fluid may penetrate.
- If the mesh is too fine, velocity interpolations become less accurate. Simulations may even crash eventually.
- The same holds if the mesh is too irregular.

A safe way is to specify a mesh with an average node-to-node distance comparable to the lattice spacing Δx. Finding an appropriate mesh for a 2D surface immersed in 3D space can be challenging and is highly problem-specific [14]!
The deformability of the membrane also plays a role.

- If the membrane is too rigid, oscillations may arise. The reason is that the standard IB-LBM is an explicit method. When the forces are too large, the algorithm becomes unstable.
- If the membrane is too deformable, local deformations may become so large that numerical problems arise. A bending resistance is often required to avoid buckling. A rule of thump is that the minimum curvature radius is limited by the mesh resolution: For higher deformations, higher resolutions are required.

Due to the finite-range interpolations, the flow field around the membrane may deviate from the expected, analytical flow field. Typically, the hydrodynamic radius of the particles is slightly larger than the input radius (about $0.3 - 0.5\Delta x$, depending on the particular case and the interpolation stencil) [14].

4 IBM for rigid objects

It is often desired to simulate rigid walls of non-trivial shape. The simplest application is a rigid cylinder in channel flow. The advantage of the IBM is that it is implemented on top of the Navier-Stokes solver without changing its algorithm. For 2D applications, it is comparably easy to create a line mesh for a complex geometry.
There are several approaches within the IB-LBM to implement rigid or quasi-rigid objects. Quasi-rigid means that the object is slightly deformable but sufficiently rigid for its purpose. It is not possible to review all of these methods in this lecture.
The quasi-rigid method by Feng and Michaelides [8] and the implicit method introduced by Wu and Shu [12] are briefly described in the following.
The so-called direct forcing method where the force is explicitly computed based on the flow field (without an additional user parameter) is not covered here. Instead, the reader should consult references like [12] and [15].
It has to be stressed that the IBM reveals its strength especially in problem where the particles are deformable. Simulating rigid particles with the IBM usually requires more input and brain work by the user.

4.1 Quasi-rigid objects

The algorithm is basically the same as for the deformable particles. The main difference is that the force is computed based on another approach. If the quasi-rigid wall is stationary, each of its surface points has a fixed reference position $\boldsymbol{x}_i^{\mathrm{ref}}(t)$. Each point is allowed to move away from its reference position, but there is a 'penalty' force of the form

$$\boldsymbol{F}_i(t) = -k\left(\boldsymbol{x}_i(t) - \boldsymbol{x}_i^{\mathrm{ref}}(t)\right). \tag{11}$$

The stiffness parameter k must be large enough so that the displacement $\boldsymbol{x}_i(t) - \boldsymbol{x}_i^{\mathrm{ref}}(t)$ remains sufficiently small.
It is clear that the value of k is critical: If it is too small, the object may deform too strongly. If it is too large, the simulation may become unstable due to the explicit nature of the algorithm.

4.2 Implicit IBM for rigid objects

Within the conventional IBM for rigid objects, the forcing term is computed explicitly in advance. However, the no-slip condition at the walls cannot be guaranteed, and streamline penetration may be observed. In order to overcome this drawback, Wu and Shu [12] introduced an implicit IBM for rigid walls in which the forcing term is treated as an unknown. The question is:

> How to choose the IBM force in such a way that the velocity of the object nodes is exactly the desired velocity?

This method has been further used by Wu et al. [16].

4.2.1 Implicit correction

The approach starts with Guo's forcing scheme [17]. The physical velocity at a fluid node is

$$\boldsymbol{u}(\boldsymbol{X},t) = \boldsymbol{u}^*(\boldsymbol{X},t) + \delta\boldsymbol{u}(\boldsymbol{X},t) \tag{12}$$

where

$$\boldsymbol{u}^* = \frac{1}{\rho}\sum_i f_i \boldsymbol{c}_i \tag{13}$$

is the velocity obtained from the first moment of the populations and

$$\delta\boldsymbol{u} = \frac{\Delta t}{2\rho}\boldsymbol{F} \tag{14}$$

is the correction due to the discrete lattice effect. The force density $\boldsymbol{F}(\boldsymbol{X},t)$ is not known at this point, but $\boldsymbol{u}^*(\boldsymbol{X},t)$ can be inferred from the known populations.
Exploiting eqs. (3) and (14), it is easy to see that

$$\delta\boldsymbol{u}(\boldsymbol{X},t) = \sum_i \delta\dot{\boldsymbol{x}}_i(t)\delta(\boldsymbol{x}_i(t) - \boldsymbol{X}) \tag{15}$$

when the density is assumed to be constant.
The velocity of the boundary nodes is obtained from eq. (2),

$$\dot{\boldsymbol{x}}_i(t) = \sum_{\boldsymbol{X}} \boldsymbol{u}(\boldsymbol{X},t)\delta(\boldsymbol{x}_i(t) - \boldsymbol{X}). \tag{16}$$

The combination of eqs. (12), (15), and (16) yields

$$\begin{aligned}\dot{\boldsymbol{x}}_i(t) &= \sum_{\boldsymbol{X}} \left(\boldsymbol{u}^*(\boldsymbol{X},t) + \delta\boldsymbol{u}(\boldsymbol{X},t)\right)\delta(\boldsymbol{x}_i(t) - \boldsymbol{X}) \\ &= \sum_{\boldsymbol{X}} \boldsymbol{u}^*(\boldsymbol{X},t)\delta(\boldsymbol{x}_i(t) - \boldsymbol{X}) + \sum_{\boldsymbol{X}}\sum_j \delta\dot{\boldsymbol{x}}_j(t)\delta(\boldsymbol{x}_j(t) - \boldsymbol{X})\delta(\boldsymbol{x}_i(t) - \boldsymbol{X})\end{aligned} \tag{17}$$

which has to be solved for $\delta\dot{\boldsymbol{x}}_j(t)$ which is the only set of unknown parameters.
It can be shown that eq. (17) can be written in matrix form [12]

$$\boldsymbol{A}\cdot\boldsymbol{Y} = \boldsymbol{B}. \tag{18}$$

The elements of the matrix $\boldsymbol{A}$ are functions of the node positions $\boldsymbol{x}_i$ only. This matrix can be inverted in order to solve for the unknown velocities $\delta\dot{\boldsymbol{x}}_i(t)$. When these are known, the forces in the Eulerian frame are obtained from eq. (14).

4.2.2 Algorithm overview

Based on the above approach, the simulation algorithm is the following:

1. Compute the matrix $\boldsymbol{A}$ and its inverse $\boldsymbol{A}^{-1}$ from the known node positions $\boldsymbol{x}_i(t)$.
2. Perform the LBM algorithm with the known body force density $\boldsymbol{f}(\boldsymbol{X}, t)$. In the first time step, set $\boldsymbol{f} = 0$.
3. Compute the uncorrected velocity $\boldsymbol{u}^*(\boldsymbol{X}, t+1)$ from the populations.
4. Solve the matrix equation $\boldsymbol{A} \cdot \boldsymbol{Y} = \boldsymbol{B}$ for the unknown corrections $\delta\dot{\boldsymbol{x}}_i(t+1)$.
5. Spread the velocity corrections to the Eulerian grid using eq. (15) and obtain $\delta\boldsymbol{u}(\boldsymbol{X}, t+1)$.
6. Obtain the force density $\boldsymbol{f}(\boldsymbol{X}, t+1)$ from eq. (14).
7. Update the node positions if the obstacle is moving, i.e., if $\dot{\boldsymbol{x}}_i(t+1) \neq 0$.
8. Go back to step 1 (if the obstacle is moving) or step 2 (if the obstacle is stationary) and compute the next time step.

4.2.3 Comments

The algorithm seems to be quite useful when the obstacles are not moving since the matrix $\boldsymbol{A}$ and its inverse do not have to be recomputed. It is assured that the velocity of the obstacle node equals their desired velocity. Streamline penetration is minimized. However, this method may still become unstable since there is no control over the force density $\boldsymbol{f}$ in the Eulerian frame.
If this algorithm is to be applied to moving objects, $\boldsymbol{A}$ and its inverse have to be recomputed at each time step. Additionally, a model for the translational and rotational motion of the object have to be implemented.

References

[1] X. Shan and H. Chen. Lattice Boltzmann model for simulating flows with multiple phases and components. *Phys. Rev. E*, 47(3):1815, 1993.

[2] O. Filippova and D. Hänel. Grid refinement for lattice-BGK models. *J. Comput. Phys.*, 147(1):219–228, 1998.

[3] Y. Peng and L.-S. Luo. A comparative study of immersed-boundary and interpolated bounce-back methods in LBE. *Prog. Comput. Fluid Dyn.*, 8(1–4):156–167, 2008.

[4] C.S. Peskin. *Flow patterns around heart valves: A digital computer method for solving the equations of motion.* Sue Golding Graduate Division of Medical Sciences, Albert Einstein College of Medicine, Yeshiva University, 1972.

[5] C.S. Peskin. Numerical analysis of blood flow in the heart. *J. Comput. Phys.*, 25(3):220–252, 1977.

[6] C.S. Peskin. The Immersed Boundary Method. *Acta Numerica*, 11:479–517, 2002.

[7] C.D. Eggleton and A.S. Popel. Large Deformation of Red Blood Cell Ghosts in a Simple Shear Flow. *Phys. Fluids*, 10(8):1834–1845, 1998.

[8] Z.-G. Feng and E.E. Michaelides. The Immersed Boundary-Lattice Boltzmann Method for Solving Fluid-Particles Interaction Problems. *J. Comput. Phys.*, 195(2):602–628, 2004.

[9] J. Zhang, P.C. Johnson, and A.S. Popel. An Immersed Boundary Lattice Boltzmann Approach to Simulate Deformable Liquid Capsules and its Application to Microscopic Blood Flows. *Phys. Biol.*, 4(4):285–295, 2007.

[10] P. Bagchi. Mesoscale Simulation of Blood Flow in Small Vessels. *Biophys. J.*, 92(6):1858–1877, 2007.

[11] M.M. Dupin, I. Halliday, C.M. Care, L. Alboul, and L.L. Munn. Modeling the Flow of Dense Suspensions of Deformable Particles in Three Dimensions. *Phys. Rev. E*, 75(6): 066707, 2007.

[12] J. Wu and C. Shu. Implicit velocity correction-based immersed boundary-lattice Boltzmann method and its applications. *J. Comput. Phys.*, 228(6):1963–1979, 2009.

[13] K. Suzuki and T. Inamuro. Effect of internal mass in the simulation of a moving body by the immersed boundary method. *Comput. Fluids*, 49(1):173–187, 2011.

[14] T. Krüger, F. Varnik, and D. Raabe. Efficient and accurate simulations of deformable particles immersed in a fluid using a combined immersed boundary lattice Boltzmann finite element method. *Comput. Math. Appl.*, 61:3485–3505, 2011.

[15] Z.-G. Feng and E.E. Michaelides. Robust treatment of no-slip boundary condition and velocity updating for the lattice-Boltzmann simulation of particulate flows. *Comput. Fluids*, 38(2):370–381, 2009.

[16] J. Wu, C. Shu, and Y.H. Zhang. Simulation of incompressible viscous flows around moving objects by a variant of immersed boundary-lattice Boltzmann method. *Int. J. Numer. Meth. Fluids*, 62(3):327–354, 2009.

[17] Z. Guo, C. Zheng, and B. Shi. Discrete Lattice Effects on the Forcing Term in the Lattice Boltzmann Method. *Phys. Rev. E*, 65(4):046308, 2002.