



# A parallel fluid–solid coupling model using *LAMMPS* and *Palabos* based on the immersed boundary method

Jifu Tan<sup>a,\*</sup>, Talid R. Sinno<sup>b</sup>, Scott L. Diamond<sup>b,\*</sup>

<sup>a</sup> Department of Mechanical Engineering, Northern Illinois University, DeKalb, IL 60115, USA

<sup>b</sup> Department of Chemical and Biomolecular Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA

## ARTICLE INFO

### Article history:

Received 31 August 2017

Received in revised form

21 December 2017

Accepted 13 February 2018

Available online 14 February 2018

### Keywords:

Lattice Boltzmann method

Palabos

LAMMPS

Immersed boundary method

Parallel computing

## ABSTRACT

The study of viscous fluid flow coupled with rigid or deformable solids has many applications in biological and engineering problems, e.g., blood cell transport, drug delivery, and particulate flow. We developed a partitioned approach to solve this coupled Multiphysics problem. The fluid motion was solved by *Palabos* (Parallel Lattice Boltzmann Solver), while the solid displacement and deformation was simulated by *LAMMPS* (Large-scale Atomic/Molecular Massively Parallel Simulator). The coupling was achieved through the immersed boundary method (IBM). The code modeled both rigid and deformable solids exposed to flow. The code was validated with the Jeffery orbits of an ellipsoid particle in shear flow, red blood cell stretching test, and effective blood viscosity flowing in tubes. It demonstrated essentially linear scaling from 512 to 8192 cores for both strong and weak scaling cases. The computing time for the coupling increased with the solid fraction. An example of the fluid–solid coupling was given for flexible filaments (drug carriers) transport in flowing blood cell suspensions, highlighting the advantages and capabilities of the developed code.

© 2018 Elsevier B.V. All rights reserved.

## 1. Introduction

Fluid flows containing solid particles are common in engineering and medicine, e.g., suspension flows [1], sedimentation [2], cell transport in blood flow [3–7], and platelet deposition on blood vessel walls [8–10]. The dynamic behavior in such phenomena is complex due to interactions between individual particles as well as interactions between the particles and the surrounding fluid and bounding walls. Moreover, the presence of highly deformable particles, such as blood cells, vesicles and polymers, make it particularly challenging to accurately describe the dynamics in such systems. Understanding the interactions between the particulate components and the fluid is essential for optimized particulate design or detailed particulate flow behavior prediction, e.g., enhance particle mixing [11], fluid coking [12], drug carrier design [13,4,14], cell separation [15], and blood clotting [10,16,17].

The present paper is focused on blood flows containing a high concentration of deformable cells that are similar in size to the vessel diameter, thus requiring explicit consideration of the particle mechanics. Readers interested in the modeling of other classes of particulate flows, e.g., those relevant to industrial applications, are

referred to other studies, e.g., Refs. [18–20]. Numerous methods have been developed to model blood flows containing cells such as red blood cells and platelets. The boundary element method is an example of a very efficient technique for these types of flows [21–23] as it formulates boundary value problems as boundary integral equations. Thus, it only requires discretization of the surface rather than the volume. However, the boundary element method requires explicit knowledge of a fundamental or analytical solution of the differential equations, e.g., linear partial differential equations. Thus, it is limited to Stokes flow conditions. The Arbitrary-Lagrangian–Eulerian method (ALE) is another approach that has been widely used to model fluid–solid interactions [24,25]. In ALE methods the fluid mesh boundary conforms to the solid boundaries on the interface, i.e., the nodes at the fluid–solid interfaces are shared. However, the ALE technique is computationally very expensive because repeated mesh generation is necessary for flows where particles experience large deformations (e.g., for red blood cells).

By contrast, non-conforming mesh methods eliminate the mesh regeneration step. In these methods, an Eulerian mesh is used for the fluid and a Lagrangian mesh is used for the solid. These two meshes are independent, i.e., they do not share nodes across the interfaces. In this case, boundary conditions are applied through imposed constraints on the interfaces. The immersed boundary method (IBM) is a popular example of a non-conforming mesh

\* Corresponding author.

E-mail addresses: [jifutan@niu.edu](mailto:jifutan@niu.edu) (J. Tan), [sld@seas.upenn.edu](mailto:sld@seas.upenn.edu) (S.L. Diamond).

technique [26,27]. Here, the velocity and force boundary conditions at the interfaces are imposed through interpolation functions that transfer velocities and forces across the domains. Consequently, the IBM approach may be regarded as an interface between two essentially separate simulators – one for evolving the particulate phase and the other for evolving the fluid. As we will show in this paper, the decoupled nature of the two components leads to a high degree of versatility in the context of software development. More generally, the present work represents an example of an emerging paradigm in multi-physics/multiscale modeling in which multiple existing (and independent) software packages are connected by a relatively simple interface to generate new functionality. Examples in the literature of such approaches include software packages for general fluid–solid coupling [28], sedimentation [29], atomic–continuum coupling [30], and fluid flow coupled with the discrete element method [19].

Another important aspect related to the implementation of methods for solving particulate flows is the portability to high performance computing (HPC) platforms so that large system sizes and long simulation times relevant to the phenomena of interest are accessible. An example of a reported HPC-enabled blood flow simulation is the work of [31], in which blood flow simulations in patient-specific coronary arteries at spatial resolutions ranging from the centimeter scale down to 10  $\mu\text{m}$  were performed on 294,912 cores. In Ref. [32], 2500 deformable red blood cells in suspension under flow were simulated on IBM Blue Gene/P supercomputers. Additional examples include large-scale simulations of blood flow in the heart [33–35], cell separation in microfluidic flows [36,15], and blood flow in the brain [37]. However, there are only a few parallel open sourced fluid–solid coupling codes, such as the immersed boundary (IB) method with support for adaptive mesh refinement (AMR) IBAMR [38], the vascular flow simulation tool *SimVascular* [39,40], the CFDEM project using computational fluid dynamics and discrete element methods [41,19]. A general open source fluid–solid coupling tool with versatile functionalities (e.g., deformable solids) and independent fluid and solid simulator is still missing.

In order to overcome challenges related to programming, model standardization, dissemination and sharing, and efficient implementation on parallel computing architectures, here we introduce a simple but effective implementation of the immersed boundary method for simulating deformable particulate flows using popular open source software. This project was also inspired by the previous work from LBDEM [41,42]. The fluid solver is based on the lattice Boltzmann method (LBM), chosen because of its efficient parallelization across multiple processor environments [43,32]. LBM is a versatile fluid flow solver engine and has been used to model diverse situations, such as flows in porous media with complex geometries [44–46] and multiphase flows [47,48]. The solid particles in the fluids, particularly, the deformable blood cells are modeled as a coarse grained cell membrane model using particle based solid solver. Many LBM based fluid–solid coupling work can be found in literature [49,50,15,51,52], some of them have not demonstrated the large scale parallel performance [51,52]. Further, none of them are open sourced yet. In the present paper we employ a general LBM fluid solver *Palabos* [53,54] and couple it with the *LAMMPS* software package [55] for describing particle dynamics. Even *LAMMPS* provides a LBM fluid solver through USER-LB extension [56,57], the functionality of the fluid solver is limited, e.g., it can only model fluid with simple geometries and apply boundary velocities on the z direction. Thus, it is still necessary to build a large-scale parallel simulation tool for fluid–solid coupling based on widely used, general purpose open source codes. The implementation of a stable, efficient large-scale parallel solver for either fluid flow or particle dynamics is nontrivial, requiring training in scientific computing and software engineering. Both

*Palabos* and *LAMMPS* are efficiently parallelized and clearly demonstrated by ample documentation and actively supported from large on-line communities: *LAMMPS* has an active mailing list that has hundreds questions and answers posted daily. These codes are also extensively validated by numerous examples and publications. Moreover, user-developers may easily extend the functionality of either package by implementing additional features, e.g., interaction potentials, integrators, fluid models, etc.

To the best of our knowledge, this is the first time *Palabos* has been coupled with *LAMMPS* in an immersed boundary method framework. The remainder of the paper is structured as follows. First, short introductions are provided in Section 2 describing the fluid solver (Section 2.1), the solid solver (Section 2.2), the immersed boundary method (Section 2.3), and the spatial decomposition for the coupling (Section 2.4). Next in Section 3.1, a validation test and convergence study are presented for a single ellipsoid in a shear flow. The parallel performance of the IBM solver is studied in Section 3.4. Finally, an example of flexible filament transport in blood cell suspensions is described in Section 3.5, highlighting the advantages and capabilities of the new solver. Conclusions and discussions are provided in Section 4.

## 2. Methods

### 2.1. Lattice Boltzmann fluid solver: Palabos

*Palabos* is an open source computational fluid dynamics solver based on the lattice Boltzmann method (LBM). It is designed in C++ with parallel features using Message Passing Interface (MPI). It has been employed widely in both academic and industrial settings. The LBM has been used extensively in blood flow modeling [58–60,10,61,46]. Reviews of the underlying theory for the LBM can be found in the literature [62–65]. LBM is usually considered as a second-order accurate method in space and time [66]. The fundamental quantity underpinning the LBM is the density distribution function  $f_i(\mathbf{x}, t)$  in phase space  $(\mathbf{x}, \vec{c}_i)$ , where  $t$  denotes the time and  $\vec{c}_i$  denotes the lattice velocity. The evolution of the density distribution function involves streaming and collision processes,

$$\underbrace{f_i(\mathbf{x} + \vec{c}_i, t + 1) - f_i(\mathbf{x}, t)}_{\text{streaming}} = \underbrace{\frac{1}{\tau}(f_i^{\text{eq}} - f_i(\mathbf{x}, t))}_{\text{BGK collision}} + F_i, \quad (1)$$

where the simplest Bhatnagar–Gross–Krook (BGK) scheme [67] is used for the collision term and  $F_i$  is the body force term that will be used to represent immersed cell boundaries [68]. The equilibrium distribution  $f_i^{\text{eq}}$  is given by

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

where  $w_i$  are the weight coefficients and  $c_s$  is the speed of sound. The density  $\rho$  and velocity  $\vec{u}$  may be calculated as

$$\rho = \sum f_i, \quad \rho \vec{u} = \sum f_i \vec{c}_i + \frac{1}{2} \vec{g}, \quad (3)$$

where  $\vec{g}$  is the external force density vector that is related to  $F_i$  as

$$F_i = \left( 1 - \frac{1}{2\tau} \right) w_i \left( \frac{\vec{c}_i - \vec{u}}{c_s^2} + \frac{\vec{c}_i \cdot \vec{u}}{c_s^4} \vec{c}_i \right) \cdot \vec{g}, \quad (4)$$

see Ref. [68] for more details on force terms. The fluid viscosity  $\nu$  is related to the relaxation parameter  $\tau$  as

$$\nu = c_s^2(\tau - 0.5) = \frac{\tau - 0.5}{3}. \quad (5)$$

In all LBM simulations reported in this paper, the fluid domain is discretized using a uniform D3Q19 lattice; see Ref. [64]. The fluid

density  $\rho$ ,  $\bar{\mathbf{u}}$  and density distribution function  $f_i$  are initialized at the equilibrium distribution calculated from Eq. (2) based on the initial fluid velocity. During each time step, streaming and collision steps are performed on  $f_i$  according to Eq. (1). Specifically, the  $f_i$  is translated in the direction of the discretized velocity vector  $\bar{\mathbf{c}}_i$  during the streaming step; then,  $f_i$  is updated based on the equilibrium distribution,  $f_i^{eq}$ , the relaxation parameter,  $\tau$ , and the force density,  $F_i$ , which is passed to the LBM from the immersed solid objects, e.g., blood cells.

## 2.2. Particle based solid solver: LAMMPS for deformable cells and particles

LAMMPS was originally designed as a molecular dynamics simulation tool [55]. In molecular dynamics, a potential function is defined to model the interactions between atoms. The force on each atom is calculated as the derivative of the potential with respect to the atomic coordinates and the atomic system motion is updated based on numerical integrations of Newton's 2nd law of motion. The LAMMPS package has now been extended to include a variety of additional dynamical engines, including peridynamics [69,70], smooth particle hydrodynamics [71,72], dissipative particle dynamics [73,74], and stochastic rotation dynamics [75]. LIGGGHTS (LAMMPS improved for general granular and granular heat transfer simulations) is also an extension of LAMMPS for discrete element method particle simulation [19]. Many predefined potentials, functions, and ODE integrators in LAMMPS make it extremely powerful for modeling atomic, soft matter [76], and biological systems [77,8].

A coarse-grained membrane model consisting of many interacting particles [78,49,79,80,46] is used to simulate red blood cells, as shown in Fig. 1a. The membrane model can bear stretching and bending. Constraints to maintain the constant membrane surface area and enclosed cell volume are imposed through harmonic potentials. The viscosity ratio of cytoplasm over blood plasma is about 5. Here we treat the blood cell internal and external viscosity to be the same for saving computational cost. Readers interested in different viscosity models are referred to Refs. [5,81,82]. The potential function for a red blood cell (RBC) used in the current work is given by

$$U(\mathbf{X}_i) = U_{stretch} + U_{bending} + U_{area} + U_{volume}, \quad (6)$$

where the stretching energy  $U_{stretch}$  is used to represent the cytoskeleton's resistance to deformation. The bending energy  $U_{bending}$  represents the rigidity of the membrane bilayer imparted to the cytoskeleton. The last two terms are the constraints for maintaining constant membrane surface area and cell volume. The stretching potential is given by:

$$U_{stretch} = \sum_{j \in 1 \dots N_s} \left[ \frac{k_B T l_m}{4p} \frac{3x_j^2 - 2x_j^3}{1 - x_j} + \frac{k_p}{l_j} \right], \quad (7)$$

where  $l_m$  is the maximum bond length, the  $j$ th bond length ratio is  $x_j = l_j / l_m$ .  $l_m$  was set to be 2 times the equilibrium bond length.  $N_s$  is the number of springs,  $p$  is the persistence length,  $k_B$  is the Boltzmann constant,  $T$  is the temperature, and  $k_p$  is the repulsive potential constant. Once  $p$  is specified,  $k_p$  may be found using the value of  $x_j = 0.5$  at the equilibrium where the net force is zero.

The bending energy is defined as

$$U_{bending} = \sum_{j=1 \dots N_s} k_b (1 - \cos(\theta_j - \theta_0)), \quad (8)$$

where  $k_b$  is the bending constant and  $\theta_j$  is the instantaneous angle formed by the two outward surface norms of two adjacent tri-

angular meshes that share the same edge  $j$ .  $\theta_0$  is the corresponding equilibrium, or spontaneous, angle.

Constraints for constant membrane surface area and cell volume are imposed through area/volume dependent harmonic potentials,

$$U_{area} = \frac{k_g (A - A_0)^2}{2A_0} + \sum_{j=1 \dots N_t} \frac{k_l (A_j - A_{j0})^2}{2A_{j0}}, \quad (9)$$

$$U_{volume} = \frac{k_v (V - V_0)^2}{2V_0}, \quad (10)$$

where  $k_g$ ,  $k_l$  are the global and local area constraint constants,  $N_t$  is the number of triangular surfaces,  $A$ ,  $A_0$  are the instantaneous and spontaneous total surface area of the cell membrane,  $A_j$ ,  $A_{j0}$  are the instantaneous and spontaneous surface area for the  $j$ th triangle surface,  $k_v$  is the volume constraint constant, and  $V$ ,  $V_0$  are the instantaneous and the equilibrium cell volume.

The parameters used in the coarse-grained membrane model can be related to membrane properties used in continuum model [79,77], e.g., the shear modulus,  $\mu_0$ , which is given by

$$\mu_0 = \frac{\sqrt{3} k_B T}{4p l_m x_0} \left[ \frac{x_0}{2(1 - x_0)^3} - \frac{1}{4(1 - x_0)^2} + \frac{1}{4} \right] + \frac{3\sqrt{3} k_p}{4l_0^3}, \quad (11)$$

where  $x_0 = l_0 / l_m$  and  $l_0$  is the bond length at equilibrium. Eq. (11) holds assuming the membrane surface area is preserved during simulation, thus the contribution from  $U_{area}$  can be safely ignored. Interested readers can refer to Refs. [79,77] for details.

For polymer particles, as shown in Fig. 1b, the stretching energy is the same as Eq. (7), while the bending energy is a harmonic function of the angle deviation,

$$U_{bending}^p = k_b^p (\theta_j - \theta_0)^2, \quad (12)$$

where the superscript  $p$  refers to polymer and the other variables are defined as in Eq. (8).

To avoid the overlapping of the particles from different solid objects, e.g., cells, polymers, etc. a Morse potential was used for inter-particle interaction,

$$U_{morse} = D_0 [e^{-2\alpha(r-r_0)} - 2e^{-\alpha(r-r_0)}], \quad r < r_c, \quad (13)$$

where  $D_0$  is the energy scale,  $\alpha$  controls the width of the potential,  $r$  is the distance between particles from different solid objects,  $r_0$  is the equilibrium distance,  $r_c$  is the cutoff distance.

All the parameters used in the simulation are listed in Table 2.

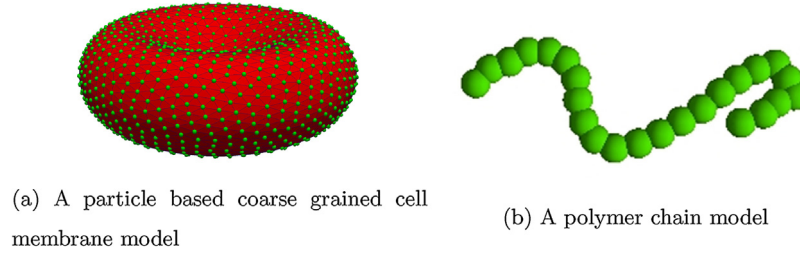
## 2.3. Fluid-solid coupling: the immersed boundary method

The immersed boundary method (IBM) was used to model the coupling between fluid and solid. The combination of LBM and IBM was first used to model fluid-particle interaction problems in Ref. [83]. Details of the immersed boundary method formulation may be found in Refs. [26,27,83]. Briefly, the solid velocity at each particle position is obtained through velocity interpolation from local fluid nodes, while fluid forces are obtained by spreading the local solid forces. Specifically, for an immersed solid with coordinates  $\mathbf{X}$ , the velocity  $\mathbf{U}(\mathbf{X}, t)$  is interpolated from the local fluid velocity  $\mathbf{u}(\mathbf{x}, t)$ , while the solid force,  $\mathbf{F}(\mathbf{X}, t)$ , calculated from Eq. (6) is spread out into the local fluid grid points as a force density  $\mathbf{f}(\mathbf{x}, t)$ :

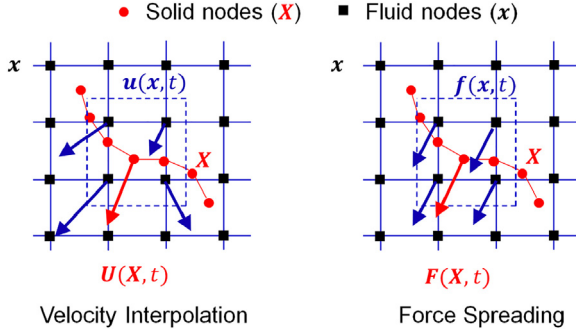
$$\mathbf{U}(\mathbf{X}, t) = \int \mathbf{u}(\mathbf{x}, t) \delta(\mathbf{x} - \mathbf{X}) d\mathbf{x}, \quad (14)$$

$$\mathbf{f}(\mathbf{x}, t) = \int \mathbf{F}(\mathbf{X}, t) \delta(\mathbf{x} - \mathbf{X}) d\mathbf{X}, \quad (15)$$

where  $\delta(\mathbf{x})$  is the delta function. In a typical numerical implementation in three dimensions,  $\delta(\mathbf{x})$  is constructed as the product



**Fig. 1.** Models for a deformable red blood cell and a polymer chain implemented in LAMMPS: (a) A particle based coarse grained red blood cell membrane model that can bear stretching and bending. The particles on the cell membrane are interacting with potentials. (b) A particle connected polymer chain model with stretching and bending resistance.



**Fig. 2.** An illustration of the fluid–solid coupling through the immersed boundary method. The solid velocity for each particle,  $\mathbf{U}(\mathbf{X}, t)$ , is interpolated from the local fluid velocities,  $\mathbf{u}(\mathbf{x}, t)$ , while the solid force at each particle  $\mathbf{F}(\mathbf{X}, t)$  is spread out onto the local fluid nodes,  $\mathbf{f}(\mathbf{x}, t)$ . The influence range of the central solid particle on fluid is shown in dashed rectangles.

of one-dimensional functions, i.e.,  $\delta(\mathbf{x}) = \phi(x)\phi(y)\phi(z)$  with  $\phi(r)$  defined as

$$\phi(r) = \begin{cases} 0, & \text{otherwise} \\ \frac{1}{4} \left( 1 + \cos\left(\frac{\pi r}{2}\right) \right), & |r| \leq 2, \end{cases} \quad (16)$$

where  $r$  is the distance between solid particles and fluid nodes. Different choices of the interpolation function influence the coupling accuracy, the influence range of the solid particles on the fluid, and the computational cost [84,85]. Schematic representations of the velocity interpolation and force spreading process used in the present implementation are shown in Fig. 2.

The coupling strategy described above becomes numerically unstable for rigid objects. In such situations, a different fluid–solid interaction (FSI) approach must be applied [86,87]. In the present IBM formulation for rigid particles, the FSI force on each particle is used to assemble the total force and torque on the whole rigid body. The translation and rotation of the rigid object are then updated based on Newton's 2nd law of motion. The FSI force can be expressed as:

$$\mathbf{f}^{FSI}(\mathbf{X}, t) = \rho_f(\mathbf{u}(\mathbf{X}, t) - \mathbf{U}(\mathbf{X}, t))/\delta t, \quad (17)$$

where  $\rho_f$  is the fluid density,  $\mathbf{u}(\mathbf{X}, t)$  is the fluid velocity at the solid boundary node  $\mathbf{X}$ , which has to be obtained through interpolation by Eq. (14).  $\mathbf{U}(\mathbf{X}, t)$  is the solid nodal velocity. The idea behind Eq. (17) is that the local fluid particles with incoming velocity  $\mathbf{u}(\mathbf{X}, t)$  will collide with the solid boundary with outgoing velocity  $\mathbf{U}(\mathbf{X}, t)$ . The FSI force is the change of the momentum divided by the collision time  $\delta t$ . Other approaches have been proposed to model rigid objects in IBM, e.g., a virtual boundary formulation was used in Ref. [88].

Another challenge in modeling fluid mediated solid transport is to correctly capture the fluid flow in the lubrication layer between two approaching solids or between one solid and a wall. For exam-

ple, when the gap between the cell membrane and the wall is very small, e.g., the gap is smaller than a lattice space, the LBM fluid solver can not resolve the fluid flow in the thin layer. One approach is to use a finer mesh for the whole fluid domain or refine the mesh near the boundary layers [89–91]. This approach would increase the LBM simulation cost as more lattice space is used. It also requires some efforts to handle the density distribution passage over the interface if a multi-grid is used in LBM. Another approach is to introduce a lubrication force to repel the cell membrane so that there are enough fluid within the gap. The lubrication repelling force was introduced to the lattice Boltzmann method in Refs. [92,93]. Following their work, the lubrication force derived from the lubrication theory between two identical spheres is

$$\mathbf{F}_{ij}^{lub} = -\frac{3\pi\nu r}{s} \hat{\mathbf{x}}_{ij} \hat{\mathbf{x}}_{ij} \cdot (\mathbf{u}_i - \mathbf{u}_j) \quad (18)$$

where  $r$  is the spherical radius,  $\nu$  is the fluid dynamic viscosity,  $s$  is dimensionless gap  $s = R/r - 2$  where  $R$  is the central distance between two spheres.  $\mathbf{x}_{ij}$  is the position vector difference between sphere  $i$  and  $j$ , defined as  $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ ,  $\hat{\mathbf{x}}_{ij}$  is the unit vector.  $\mathbf{u}$  is the spherical velocity. Eq. (18) can also be extended to the case where a sphere approach a stationary wall by setting  $\mathbf{u}_j = 0$ . Interested readers can find more details there. In the present paper, we resolve the lubrication effect using smaller fluid mesh size, e.g., the fluid mesh size was set to 100 nm when a blood cell squeezing through a 5  $\mu\text{m}$  tube, as shown in Section 3.3.

#### 2.4. Spatial decomposition for fluid–solid coupling

Spatial decomposition is adopted by both *Palabos* and *LAMMPS* for parallel computing. In our fluid–solid coupling approach, the same spatial decomposition was applied to both the fluid and the solid so that the coupling can be effectively handled by the same compute core for a given region. Consistent spatial decomposition for the two domains ensures that individual processors have access to both the fluid grid points and solid particles within the same sub-domain. An illustration of the partitioning process is shown in Fig. 3, where the whole fluid–solid system is partitioned onto 8 compute cores. Ghost layers near the boundaries of each sub-domain are used for communication between neighboring cores. This approach may not be optimal for a system where solid particles are highly heterogeneously distributed across the entire domain. However, for most cases of interest the distribution of solid particles is quite homogeneous, and in any case, the majority of the computation is dedicated to the fluid solver. The issue of load optimization for heterogeneous systems is deferred to future work.

The FSI coupling algorithm consists almost entirely of two routines to carry out velocity interpolation and force spreading functions. As mentioned above the fluid solver usually represents the bulk of the computational demand because the solid fraction is usually quite small. Based on this intrinsic asymmetry, we employed *Palabos* as the driving code, while *LAMMPS* was called



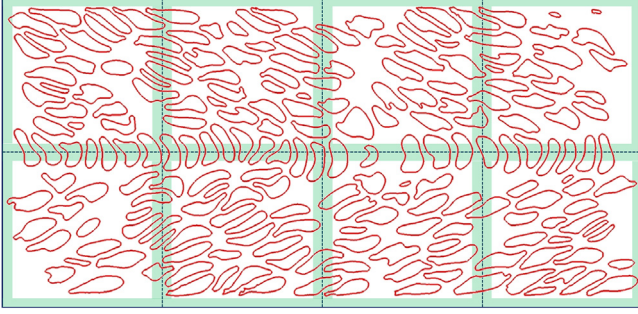
```

template<typename T,template<typename U> class Descriptor>
void interpolateVelocity3D (MultiBlockLattice3D<T,Descriptor>
    &lattice ,LAMMPSWrapper &wrapper );

template<typename T,template<typename U> class Descriptor>
void spreadForce3D (MultiBlockLattice3D<T,Descriptor>
    &lattice ,LAMMPSWrapper &wrapper );

```

List 1. Function declarations for IBM.



**Fig. 3.** Spatial decomposition for parallel computing for fluid–solid coupling. The blood flow with cells (red lines) was divided into multiple regions. One processor was assigned to each region to calculate the both fluid and solid motions within the region, thus, the coupling between fluid and solid within the region is local to the processor. On the interface between processors, ghost layers (green stripe) were used for communication for both fluid and solid independently. An example of 8 processors were used for the task partition in the figure. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

as an external library. To fully access all the members (e.g., particle positions, velocities, forces, etc.) from *LAMMPS*, a pointer to *LAMMPS* was used and passed to the two interpolation functions. An outline of the functions implemented in our IBM algorithm is shown in List 1.

Where *lattice* is the structure used in *Palabos* to store the population distribution functions for the LBM, while *wrapper* is a pointer to an instance of *LAMMPS*. The advantage of the IBM in terms of software development is that only two functions are needed to couple a fluid solver to a solid solver. Our IBM implementation therefore only requires a few hundred lines of code beyond what is implicitly contained in the *Palabos* and *LAMMPS* packages. Detailed implementations can be found at <https://github.com/TJFord/palabos-lammps>.

### 3. Results

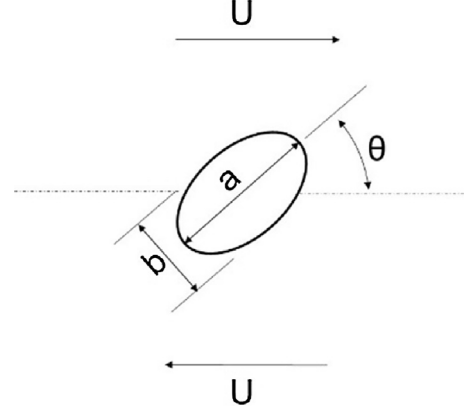
#### 3.1. Validation I: Ellipsoid in shear flow

To validate our IBM implementation, we considered the trajectory of a rigid ellipsoid in a shear flow in Stokes flow regime, commonly referred to as the Jeffery's orbit [94], where the rotation angle of the ellipsoid,  $\theta$ , satisfies

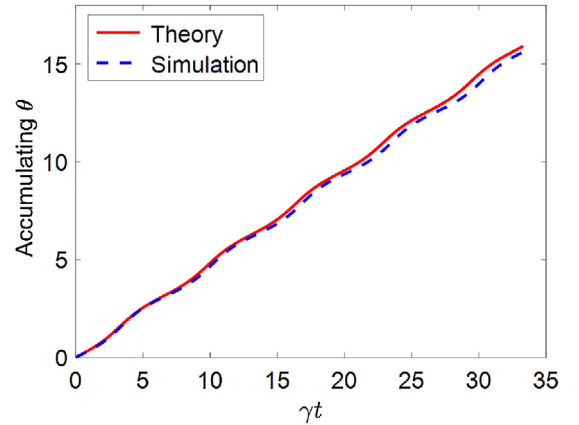
$$\tan \theta = \frac{b}{a} \tan \frac{a^2 b \gamma t}{a^2 + b^2}, \quad (19)$$

where  $\theta$  is the angle formed by the major axis of the ellipsoid and the shear flow direction,  $a, b$  are the major and minor semi-axis lengths,  $\gamma$  is the shear rate, and  $t$  is the time. A 2D illustration of the ellipsoid is shown in Fig. 4 – the third semi-axis length is assumed to be the same as  $b$ .

An ellipsoid with semi-axis lengths  $a=6, b=c=4.5$  was placed at the center of a channel with height  $H=60$ . The top and bot-



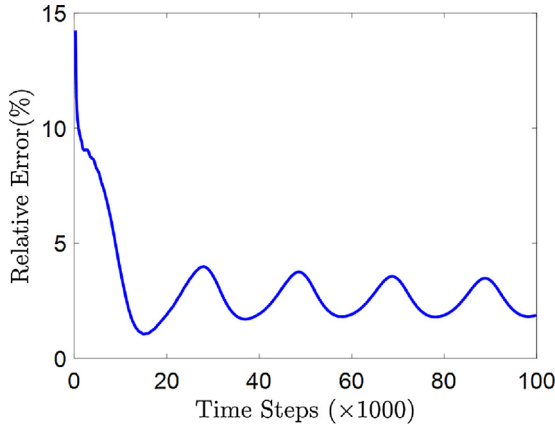
**Fig. 4.** An illustration of an ellipsoid in a shear flow.  $a, b$  are the major and minor semi-axis lengths, and  $U$  is the magnitude of the velocities applied at the top and bottom plates. The distance between the top and bottom parallel plate is  $H$ , giving a shear rate  $\gamma = 2U/H$ .



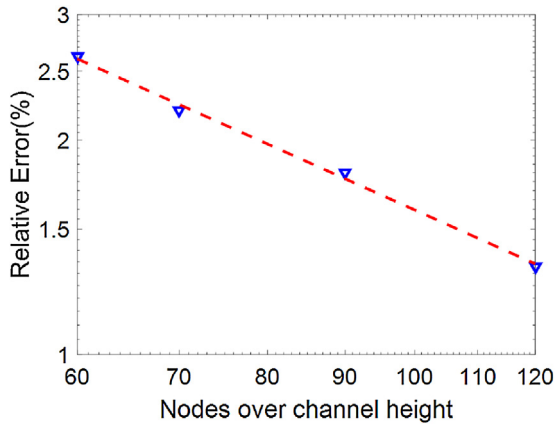
**Fig. 5.** The comparison between Jeffery's orbit given by Eq. (19) and numerical results simulated by the fluid–solid coupling code.

tom velocities were set to  $U=0.01$  and a viscosity of  $\nu=1/6$  was prescribed. These parameters give a shear rate  $\gamma=0.00033$  and a Reynolds number of  $Re=aU/\nu=0.36$  to approximate Stokes flow. The IBM numerical result for the orientation angle  $\theta$  and the analytical solution given by Eq. (19) are plotted in Fig. 5. The agreement is good, validating the FSI coupling. Shown in Fig. 6 is the relative error, defined as  $\frac{|\theta_{sim}-\theta_{the}|}{\theta_{the}}$  where  $\theta_{sim}$  is the simulation data and  $\theta_{the}$  is the theoretical data from Eq. (19). The relative error is seen to oscillate with the period of the ellipsoid and is, on average, constant at about 2.62%.

As expected, the relative error  $\epsilon$  is a power function of the fluid grid resolution  $N$  – Fig. 7 shows that the relative error scales as the grid spacing following  $\epsilon \propto N^{-0.96}$ .



**Fig. 6.** The relative error of the ellipsoid rotation angle  $\theta$  calculated based on the fluid–solid coupling at  $Re = 0.36$ . The fluid grid was  $60 \times 60 \times 30$ .



**Fig. 7.** The convergence for the fluid–solid coupling at  $Re = 0.36$  with different grid resolutions. The convergence rate is nearly linear to the grid spacing, as indicated by the dashed line. The relative error  $\epsilon \propto N^{-0.96}$  where  $N$  is the grid resolution over channel height  $H$ .

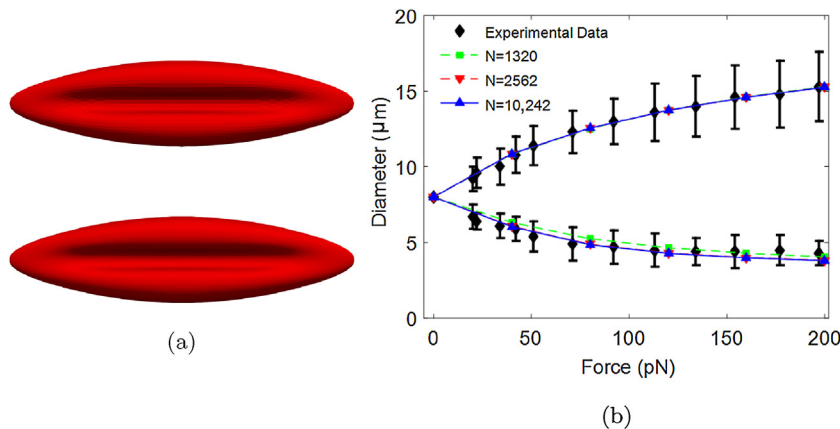
### 3.2. Validation II: Red blood cell stretching test

The RBC model was also validated and compared with optical tweezer stretching results under different loadings. The cell diameter without deformation was  $8 \mu\text{m}$ , bending stiffness  $k_b = 2.3 \times 10^{-19} \text{ J}$ ,  $k_g = k_l = 2.1 \times 10^{-4} \text{ N/m}$ ,  $k_v = 2.2 \text{ N/m}^2$ . The

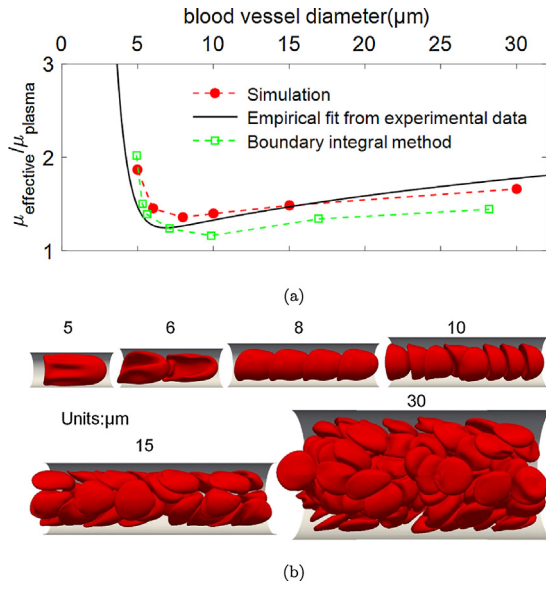
stretching constant  $k_B T/p$  depended on the number of discretized membrane particles, e.g.,  $k_B T/p = 1 \times 10^{-12} \text{ N}$  for 1320 surface particles,  $k_B T/p = 7.14 \times 10^{-13} \text{ N}$  for 2562 surface particles,  $k_B T/p = 3.57 \times 10^{-13} \text{ N}$  for 10,242 surface particles. However, the shear modulus  $\mu_0$  was kept constant with  $\mu_0 = 6 \mu\text{N/m}$  for all the meshes. A pair of forces was applied to two membrane patches on each side, occupying about 5% of the total number of particles. The applied force was shared uniformly among these particles. Velocity verlet algorithm was used to integrate the Newton's equation. Viscous damping was used to stabilize the system. The deformation of RBCs under stretching (Fig. 8a), particularly, the axial and transversal diameter (Fig. 8b), were recorded and shown in Fig. 8. In Fig. 8b, the top curve was the axial diameter, and the bottom curve was the transversal diameter. Both curves agreed well with experimental optical tweezer stretching data [79] for three different mesh sizes, validating the coarse grained molecular dynamics cell model.

### 3.3. Validation III: Effective blood viscosity

Blood viscosity changes as it flows through different tube diameters and hematocrits. The viscosity decreases as the tube diameter in the range of  $10\text{--}300 \mu\text{m}$  due to the Fahraus–Lindqvist effect [95]. The effective viscosity of blood in tube flow was studied and compared with experimental results [96]. The hematocrit defined as the actual volume ratio between cells and the fluid was 30%. The tube diameter was ranging from  $5 \mu\text{m}$  to  $30 \mu\text{m}$ . The fluid mesh size and the number of cell membrane nodes were  $100 \text{ nm}$  and  $10,242$  for  $5$  and  $6 \mu\text{m}$  tube,  $200 \text{ nm}$  and  $2562$  for  $8, 10, 15 \mu\text{m}$ ,  $333 \text{ nm}$  and  $2562$  for  $30 \mu\text{m}$ . The fluid mesh size was selected in a way such that it can resolve the fluid flow between cells and the tube wall. The mechanical properties were exact the same, as shown in Section 3.2. A pressure gradient was applied to drive the flow. The Reynold's number defined with cell diameter and average flow speed was around  $0.017 \ll 1$ . Periodic boundary conditions were applied at the inlet and outlet. The simulation ran for enough time to reach quasi-steady state, e.g., the volume rate was steady. The effective viscosity was normalized with respect to the plasma viscosity, and shown in Fig. 9. The simulation data clearly showed a decreasing viscosity as the tube diameter decreased from  $30$  to  $8 \mu\text{m}$ . The viscosity increased as the tube diameter decreased further, which was resulted from the physical membrane contact with the wall. The viscosity curve agreed well with the empirical fitting from Ref. [96], which validated that the model can capture the collective blood cell behavior under flow. The effective viscosity simulated by bound-



**Fig. 8.** (a) The deformed shape of a red blood cell under  $200 \text{ pN}$  using  $2562$  membrane particles (top) and  $10,242$  membrane particles (bottom). (b) Comparison between the simulation results and optical tweezer experimental results [79] for the force diameter curve for red blood cells under different loadings. Simulation results were presented with three different membrane mesh size, e.g.,  $N = 1320, 2562$ , and  $10,242$ , respectively.



**Fig. 9.** (a) The normalized effective viscosity for blood flowing through tubes with different diameters. The data by boundary integral method was from Ref. [97], the empirical fit curve was from Ref. [96]. (b) The snapshots for the red blood cells flowing through tubes with different diameters, showing different red blood cell shapes under flow. The hematocrit was 30%.

ary integral method [97] was also provided for comparison. Both methods can predict the blood viscosity reasonably well.

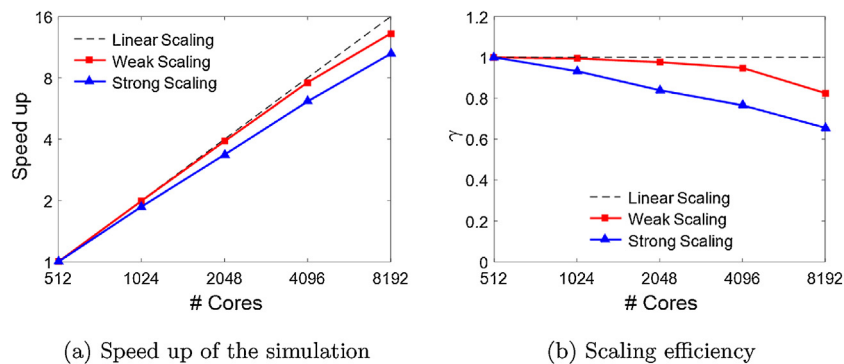
#### 3.4. Scalability: parallel performance

The parallel performance of the code was examined using a benchmark simulation of red blood cells flowing in a rectangular box. The fluid domain was discretized into a  $360 \times 360 \times 720$  grid in  $x, y, z$  directions with one inlet and one outlet and 4 nonslip side walls. The fluid flow was driven by a body force of  $1.34 \times 10^{-7}$  with periodic inlet and outlet boundary conditions in the  $z$  direction. The solid phase consisted of 15,680 red blood cells, discretized into 20,697,600 particles connected by 61,998,720 bonds, 41,332,480 angles, and 61,998,720 dihedrals in total. The hematocrit was about 40.8%. The cells were initially uniformly distributed in the flow domain. The simulations were executed on the IBM Blue Gene/Q system from Argonne National Laboratory. A ghost layer with thickness of three fluid lattice spacings was employed for inter-processor communication within the fluid solver. The ghost cutoff distance for LAMMPS was set to 1.5 times the LBM lattice spacing. The CPU time for 100 time steps was recorded. Both strong scaling and weak scaling were considered. For weak scaling cases, the

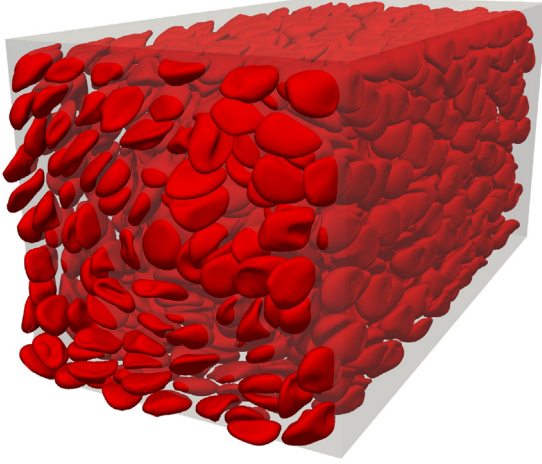
fluid domain size increased from  $360 \times 360 \times 45$  for 512 cores to  $360 \times 360 \times 720$  for 8192 cores. The parallelization performance was analyzed using two parameters: speed up ( $SU$ ) and scaling efficiency ( $\gamma$ ). The speed up and scaling efficiency were calculated as  $SU(N) = \frac{t_0^s}{t_N^s}$ ,  $\gamma(N) = \frac{t_0^s N_0}{t_N^s N}$  for strong scaling cases, and  $SU(N) = \frac{t_0^w N_0}{t_N^w N}$ ,  $\gamma(N) = \frac{t_0^w}{t_N^w}$  for weak scaling cases, where  $t_0$  was the reference simulation time for  $N_0$  cores,  $t_N$  was the simulation time for  $N$  cores. The subscript  $s, w$  denoted strong and weak scaling cases, respectively. Here  $N_0$  was taken as 512. The speed up of the simulation time as well as scaling efficiency were shown in Fig. 10. It showed that, as the number of cores increased from 512 to 8192, the speed up increased from 1 to 10.5 for strong scaling case, from 1 to 13.2 for weak scaling cases, see Fig. 10a. Similarly, the scaling efficiency dropped from 1 to 0.656 for strong scaling case, and from 1 to 0.824 for weak scaling case, see Fig. 10b. The dashed lines showed the ideal cases with no overhead and communication cost. It demonstrated that the combination of *Palabos* and *LAMMPS* using the immersed boundary method can achieve high performance among thousands of processors.

The scaling efficiency was found to depend strongly on the computer hardware. For example, less parallel efficiency was obtained on another computer with Intel Xeon E5630 2.53 GHz processors (16 cores in total) compared with 2 AMD Opteron 8-core 6128 processors with a clock speed of 2.0 GHz. While we did not study architecture dependence in more detail here, these results suggest that further work is needed to assess architecture-dependent performance issues. Finally, we note that the ghost layer thickness also influenced the performance of the IBM code. In the present study we used 3 layers of lattice points, while other studies have employed linear interpolation kernels that only require a single layer of lattice points for the communication layer [98].

The extra computational time induced by the coupling can be analyzed by GNU profiler tools. We considered the blood flow in a microfluidic channel with size of  $60 \times 60 \times 120 \mu\text{m}$ . 1470 cells were placed in the channel, resulting a hematocrit of 31% in the flow. In total there were 1,940,400 cell membrane particles and 11,664,000 fluid lattices, as shown in Fig. 11. On average, the extra computational time required for the information exchange between the fluid and solid solvers was about 34%, e.g., time spent on the two coupling functions: *interpolateVelocity3D* and *spreadForce3D*. The computational time for the two coupling functions expressed in terms of the percentage over the total CPU time is shown in Table 1. The function *interpolateVelocity3D* cost about 9% on average, while function *spreadForce3D* cost about 25%. The percentage for both functions decreased as more cores were used. This was due to the increased communication cost among many processors. As expected, the coupling time would increase as the hematocrit



**Fig. 10.** (a) The speed up of the simulation for strong and weak scaling cases. (b) The strong and weak scaling efficiency of the code based on the immersed boundary method. The dashed lines showed the ideal linear scaling performance. All the simulations were performed on the IBM Blue Gene/Q system from Argonne National Laboratory.



**Fig. 11.** A snapshot of the 1470 blood cells moving in a microfluidic channel with a size of  $60 \times 60 \times 120 \mu\text{m}$ . The fluid domain was discretized into 11,664,000 lattices, the cells were discretized into 1,940,400 membrane nodes. The hematocrit level was 30%.

**Table 1**  
The percentage of CPU time for the coupling functions.

ID	Functions	CPU time (%)				
		p1	p2	p4	p8	p16
$t_1$	interpolateVelocity3D	10.12	10.86	10.07	8.72	7.12
$t_2$	spreadForce3D	28.25	27.27	26.45	23.75	18.23
$t_1 + t_2$		38.37	38.12	36.51	32.46	25.35

level, as the interpolation and spreading was done for each cell membrane particle.

### 3.5. Case study: transport of flexible filaments in flowing red blood cell suspensions

Next, the validated IBM code was applied to a problem relevant to drug carrier delivery in microcirculation flows to demonstrate its capabilities in a more complex setting. This is a multi-physics modeling problem as it involves fluid flow, large cell deformations, and polymer transport [6,4,99,4,7]. Of particular interest is the observation that particles with different sizes, shapes, or flexibility can exhibit distinct transport properties in the blood stream. For example, long flexible filaments persist in the circulation up to one week after intravenous injection in rodent models. This is about ten times longer than for spherical counterparts [100]. The flexibility and filament length may influence transport and contribute to the anomalously long circulation time.

We simulated filament transport in tube flow of red blood cell suspensions and analyzed filament migration, apparent size, spatial distribution, and dispersion rate. The fluid domain was represented by a cylindrical tube with diameter of  $30 \mu\text{m}$  and length  $50 \mu\text{m}$ , as shown in Fig. 12. No-slip conditions were applied at the cylinder wall while periodic boundary conditions were applied at the inlet and outlet for both fluid and cells. The relaxation time was set to  $\tau = 1$  giving a dimensionless kinetic viscosity of  $1/6$ . The LBM lattice spacing,  $dx$ , was fixed at  $0.33 \mu\text{m}$ , the time step was set to  $dt = 1.82 \times 10^{-8} \text{ s}$ . One hundred red blood cells were placed in the domain corresponding to a cell concentration of about 27%, which is within the physiological range [101]. Each cell was described by 1320 surface nodes, 3954 stretching bonds, 2636 area segments, and 3954 bending angles. A force density was applied to drive the flow such that the shear rate at the wall was  $1000 \text{ s}^{-1}$  without the red blood cells. The actual wall shear rate was less as the blood cell component reduces the flow rate. A total of 156 filaments were

**Table 2**

Simulation parameters used for the particle transport in blood flow. The shear modulus for the cell  $\mu_0 = 6 \mu\text{N/m}$ . The stretching and bending parameters for filaments were selected based on [102,103]. The filaments were assumed to be straight in equilibrium, and the equilibrium length was chosen due to the resolution of the fluid lattice. The Morse potential was used to avoid the filament-cell and cell-cell overlapping.

Red blood cells				
$k_B T/p$ (N)	$k_b$ (J)	$k_g$ (N/m)	$k_l$ (N/m)	$k_v$ (N/m <sup>2</sup> )
$1 \times 10^{-12}$	$2.3 \times 10^{-19}$	$2.1 \times 10^{-4}$	$2.1 \times 10^{-4}$	2.2
Filaments				
$k_B T/p$ (N)	$k_p$ (J)	$k_b$ (J)	$l_0$ ( $\mu\text{m}$ )	$\theta_0$ ( $^\circ$ )
$1.36 \times 10^{-12}$	$1.7 \times 10^{-12}$	$[4.1, 410] \times 10^{-20}$	0.33	180
Morse potential				
$D_0$ (J)	$\alpha$	$r_0$ ( $\mu\text{m}$ )	$r_c$ ( $\mu\text{m}$ )	
$1.2 \times 10^{-18}$	0.5	0.66	0.66	

then introduced at one end of the cylinder as shown in Fig. 12. Each filament was a polymer chain modeled as beads connected by stretching and bending springs. Different lengths ( $2 \mu\text{m}$  and  $8 \mu\text{m}$ ) and different bending stiffnesses ( $4.1 \times 10^{-20} \text{ J}$ , or about  $10k_B T$ , and  $4.1 \times 10^{-18} \text{ J}$ , corresponding to about  $1000k_B T$ ) were used to study the size and stiffness effect on filament transport in red blood cell suspensions.

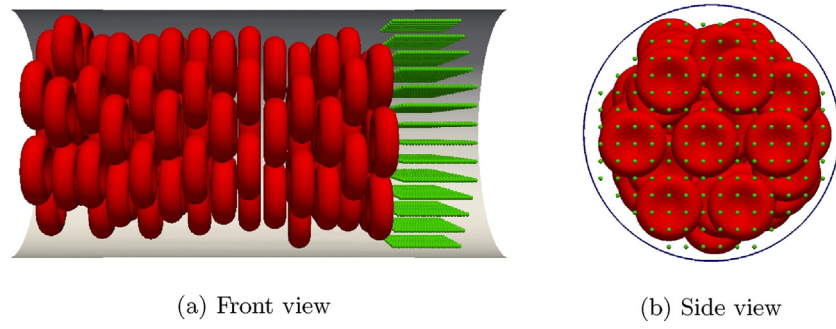
The parameters used in the simulations are listed in Table 2. The corresponding shear modulus for red blood cell membranes was  $\mu_0 = 6 \mu\text{N/m}$  based on Eq. (11).

Simulation snapshots at  $8 \times 10^6$  time steps are shown in Fig. 13 for cases with different filament size and flexibility. Different configurations are observed for filaments with different stiffnesses. The stiffer filaments were generally straight when the length was  $2 \mu\text{m}$ , and exhibited a small amount of curvature for  $8 \mu\text{m}$ , see Fig. 13b and d. By contrast, the more flexible filaments exhibited highly bent or coiled shapes, particularly for the  $8 \mu\text{m}$  filaments, see Fig. 13a and c. The apparent filament size, defined as the maximum extent for the filaments, is shown in Table 3. The apparent size for stiffer filaments was very close to the actual contour length, while the apparent size for flexible filaments was smaller than the actual contour length, due to the bending and coiling of the filaments.

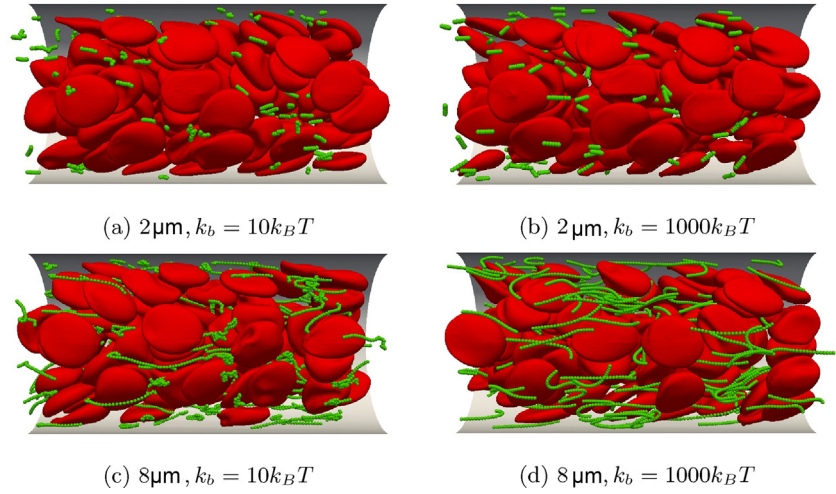
The average radial position  $\langle r \rangle$  normalized to the vessel radius  $R$  and the mean square displacement (MSD) for the filaments are plotted in Fig. 14a and b. Filament migration toward the vessel wall was observed for all filament types, as indicated by the increasing values of  $\langle r \rangle/R$ . The averaged mean radial positions for flexible filaments ( $10k_B T$ ) was stabilized around  $0.7R$  for both sizes. For stiffer filaments ( $1000k_B T$ ), the  $\langle r \rangle/R$  kept increasing within the simulation time, showing better margination properties than flexible ones. This agrees well with previous findings for deformable particles [13]. Less margination for flexible ones lead to low binding to the vessel wall and high concentration in the blood stream, which can also explain the long circulation time for long flexible filaments in rodents [100]. The dispersion rate, defined as the slope of the MSD curve, is shown in Table 3. The dispersion rates are all of the order of  $10^{-11} \text{ m}^2/\text{s}$ , which is about 2 orders-of-magnitude larger than the background thermal diffusion expected for particles of this size. This finding is also consistent with other observations for small particles in blood suspensions, such as platelets [23,104,105], particles [4,13] and experimental measurements for microparticles [106].

The fraction of filaments present in the  $3 \mu\text{m}$  cell-free layer (CFL) near the vessel wall is shown in Fig. 15. Only the data in the last  $0.15 \text{ s}$  were selected for analysis due to the transient effect in the beginning. The fraction of the  $2 \mu\text{m}$  filaments in the CFL increases





**Fig. 12.** Initial positions for the blood cells and filaments. The cells were randomly distributed in the blood vessel, while the filaments were uniformly distributed in one end of the vessel.

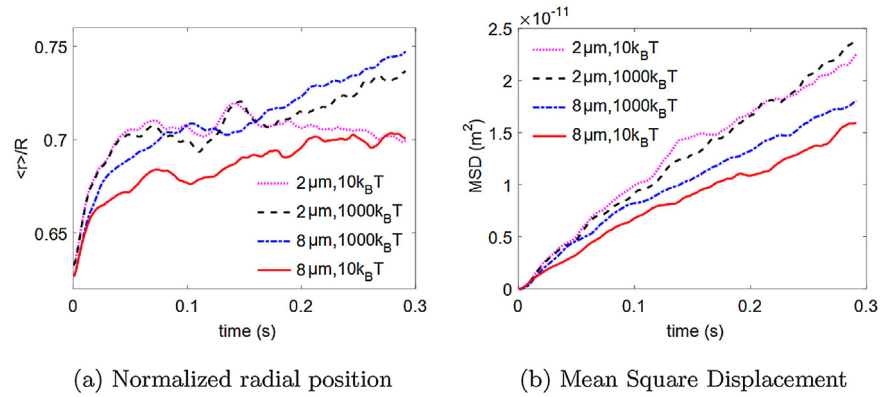


**Fig. 13.** Snapshots of the simulations of filaments (green) mixing with blood cell suspensions (red) at  $8 \times 10^6$  time steps.  $2\mu\text{m}$  filaments with bending stiffness  $10k_B T$  (a) and  $1000k_B T$  (b);  $8\mu\text{m}$  filaments with bending stiffness  $10k_B T$  (c) and  $1000k_B T$  (d). Half of the vessel wall was also shown in the figure. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

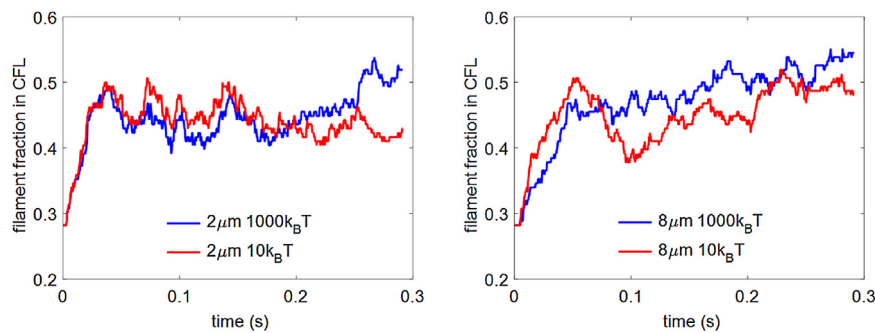
**Table 3**

The apparent size and dispersion rate of the filaments in the blood flow with cells. The apparent size of the filaments was smaller than the actual contour length due to the bending and coiling. The dispersion rate was in the order of  $10^{-11} \text{ m}^2/\text{s}$ , which was about 2 orders-of-magnitude larger than thermal diffusion for microparticles.

Contour size ( $\mu\text{m}$ )	1.66		7.92	
Stiffness ( $k_b/k_B T$ )	10	1000	10	1000
Apparent size ( $\mu\text{m}$ )	$1.57 \pm 0.31$	$1.67 \pm 0.01$	$5.76 \pm 2.4$	$7.89 \pm 1.92$
Dispersion rate ( $\times 10^{-11} \text{ m}^2/\text{s}$ )	9.76	8.64	6.26	7.26



**Fig. 14.** (a) The averaged filament center position normalized with the vessel radius  $R = 15 \mu\text{m}$ . Stiffer filaments showed better margination properties than flexible filaments. (b) The mean square displacement for the filaments transported with the blood cells.



**Fig. 15.** The fraction of the filaments in the cell free layer of the blood vessel. The cell free layer was defined as a 3  $\mu\text{m}$  thickness layer next to the vessel wall. (a) 2  $\mu\text{m}$ ; (b) 8  $\mu\text{m}$ .

approximately from 28% (the baseline value assuming a uniform distribution) to  $43.3\% \pm 1.7\%$  for soft filaments, and to  $46.1\% \pm 3.4\%$  for stiff filaments. However, the fraction of 8  $\mu\text{m}$  filaments in the CFL reached  $47.5\% \pm 2.4\%$  for the flexible ones, and  $51.1\% \pm 1.9\%$  for stiffer ones. These results suggest that longer and stiffer filaments in blood cell suspensions margined quickly than the shorter and softer ones. The margination for rigid filaments also required less traveling distance in the flow direction, as the effective viscosity was increased for the mixture of rigid filaments in blood cell suspensions.

#### 4. Conclusions

We introduced an efficient implementation of the Immersed Boundary Method by coupling a Lattice Boltzmann fluid solver with a particle-based solid solver using open source codes, namely *Palabos* and *LAMMPS*, respectively. The coupling was achieved by a simple interface that only requires a few hundred lines of code, dramatically reducing software development time, increasing robustness and facilitating transferability. The coupling was demonstrated to be computationally efficient. The extra computing time depends on the solid fraction, e.g., for a hematocrit of 30%, it required only about a 34% increase in the computational cost beyond what was required for executing fluid and solid solvers. The IBM code was also shown to scale linearly from 512 to 8192 cores for both strong and weak scaling cases, demonstrating great parallelization efficiency in massively parallel environments.

The validated IBM code was used to analyze polymer filament transport in red blood cell flows; such filaments are of interest for drug delivery applications. Simulations demonstrated how filament flexibility reduce the effective size of filament, so that flexible filaments can traverse through the blood cell suspensions easier than stiffer ones. The stiffer filaments showed better margination properties than flexible ones, consistent with previous findings [13]. These results highlight the complex physics that may underlie the long circulation time found for such particles in rodents [100].

#### Acknowledgments

This work was supported by NIH1U01HL131053-01A1. We thank the high performance computing support from Gaea at Northern Illinois University and the Mira at the Argonne National Laboratory.

#### References

- [1] P.R. Nott, J.F. Brady, Pressure-driven flow of suspensions: simulation and theory, *J. Fluid Mech.* 275 (1994) 157–199.
- [2] J. Feng, H.H. Hu, D.D. Joseph, Direct simulation of initial value problems for the motion of solid bodies in a Newtonian fluid. Part 1. Sedimentation, *J. Fluid Mech.* 261 (1994) 95–134.
- [3] D.A. Fedosov, B. Caswell, A.S. Popel, G.E. Karniadakis, Blood flow and cell-free layer in microvessels, *Microcirculation* 17 (8) (2010) 615–628.
- [4] K. Vahidkhah, P. Bagchi, Microparticle shape effects on margination, near-wall dynamics and adhesion in a three-dimensional simulation of red blood cell suspension, *Soft Matter* 11 (11) (2015) 2097–2109.
- [5] S.K. Dodd, P. Bagchi, Lateral migration of a capsule in a plane Poiseuille flow in a channel, *Int. J. Multiph. Flow* 34 (10) (2008) 966–986.
- [6] K. Sinha, M.D. Graham, Shape-mediated margination and demargination in flowing multicomponent suspensions of deformable capsules, *Soft Matter* 12 (6) (2016) 1683–1700.
- [7] J. Tan, A. Thomas, Y. Liu, Influence of red blood cells on nanoparticle targeted delivery in microcirculation, *Soft Matter* 8 (6) (2012) 1934–1946.
- [8] I.V. Pivkin, P.D. Richardson, G. Karniadakis, Blood flow velocity effects and role of activation delay time on growth and form of platelet thrombi, *Proc. Natl. Acad. Sci. U. S. A.* 103 (46) (2006) 17164–17169.
- [9] A.L. Fogelson, R.D. Guy, Immersed-boundary-type models of intravascular platelet aggregation, *Comput. Methods Appl. Mech. Eng.* 197 (25) (2008) 2087–2104.
- [10] Y. Lu, M.Y. Lee, S. Zhu, T. Sinno, S.L. Diamond, Multiscale simulation of thrombus growth and vessel occlusion triggered by collagen/tissue factor using a data-driven model of combinatorial platelet signalling, *Math. Med. Biol.* (2016) dqw015.
- [11] Y. Tsuji, T. Kawaguchi, T. Tanaka, Discrete particle simulation of two-dimensional fluidized bed, *Powder Technol.* 77 (1) (1993) 79–87.
- [12] J. Gao, J. Chang, C. Xu, X. Lan, Y. Yang, CFD simulation of gas solid flow in FCC strippers, *Chem. Eng. Sci.* 63 (7) (2008) 1827–1841.
- [13] K. Müller, D.A. Fedosov, G. Gompper, Understanding particle margination in blood flow – a step toward optimized drug delivery systems, *Med. Eng. Phys.* 38 (1) (2016) 2–10.
- [14] R. Radhakrishnan, B. Uma, J. Liu, P.S. Ayyaswamy, D.M. Eckmann, Temporal multiscale approach for nanocarrier motion with simultaneous adhesion and hydrodynamic interactions in targeted drug delivery, *J. Comput. Phys.* 244 (2013) 252–263.
- [15] T. Krüger, D. Holmes, P.V. Coveney, Deformability-based red blood cell separation in deterministic lateral displacement devices – a simulation study, *Biomicrofluidics* 8 (5) (2014) 054114.
- [16] A. Yazdani, H. Li, J.D. Humphrey, G.E. Karniadakis, A general shear-dependent model for thrombus formation, *PLoS Comput. Biol.* 13 (1) (2017) e1005291.
- [17] A.L. Fogelson, K.B. Nieves, Fluid mechanics of blood clot formation, *Annu. Rev. Fluid Mech.* 47 (2015) 377–403.
- [18] H. Zhu, Z. Zhou, R. Yang, A. Yu, Discrete particle simulation of particulate systems: a review of major applications and findings, *Chem. Eng. Sci.* 63 (23) (2008) 5728–5770.
- [19] C. Kloss, C. Goniva, A. Hager, S. Amberger, S. Pirker, Models, algorithms and validation for opensource DEM and CFD-DEM, *Prog. Comput. Fluid Dyn.* 12 (2–3) (2012) 140–152.
- [20] M. Chiesa, V. Mathiesen, J.A. Melheim, B. Halvorsen, Numerical simulation of particulate flow by the Eulerian–Lagrangian and the Eulerian–Eulerian approach with application to a fluidized bed, *Comput. Chem. Eng.* 29 (2) (2005) 291–304.
- [21] J.B. Freund, The flow of red blood cells through a narrow spleen-like slit, *Phys. Fluids* (1994–present) 25 (11) (2013) 110807.
- [22] C. Pozrikidis, Effect of membrane bending stiffness on the deformation of capsules in simple shear flow, *J. Fluid Mech.* 440 (2001) 269–291.
- [23] H. Zhao, E.S. Shaqfeh, V. Narsimhan, Shear-induced particle migration and margination in a cellular suspension, *Phys. Fluids* (1994–present) 24 (1) (2012) 011902.
- [24] J. Donea, S. Giuliani, J.-P. Halleux, An arbitrary Lagrangian–Eulerian finite element method for transient dynamic fluid–structure interactions, *Comput. Methods Appl. Mech. Eng.* 33 (1–3) (1982) 689–723.
- [25] M. Souli, A. Ouahsine, L. Lewin, Ale formulation for fluid–structure interaction problems, *Comput. Methods Appl. Mech. Eng.* 190 (5) (2000) 659–675.
- [26] C.S. Peskin, The immersed boundary method, *Acta Numer.* 11 (2002) 479–517.

- [27] R. Mittal, G. Iaccarino, Immersed boundary methods, *Annu. Rev. Fluid Mech.* 37 (2005) 239–261.
- [28] E. Constant, C. Li, J. Favier, M. Meldi, P. Meliga, E. Serre, Implementation of a Discrete Immersed Boundary Method in Openfoam, 2016 arXiv:1609.04364.
- [29] R. Sun, H. Xiao, Sedifoam: A general-purpose, open-source CFD-DEM solver for particle-laden flow with emphasis on sediment transport, *Comput. Geosci.* 89 (2016) 207–219.
- [30] I.A. Couden, J.R. Lukes, A hybrid atomistic–continuum model for fluid flow using LAMMPS and OpenFOAM, *Comput. Phys. Commun.* 184 (8) (2013) 1958–1965.
- [31] A. Peters, S. Melchionna, E. Kaxiras, J. Lätt, J. Sircar, M. Bernaschi, M. Bison, S. Succi, Multiscale simulation of cardiovascular flows on the IBM BlueGene/P: full heart-circulation system at red-blood cell resolution, in: *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE Computer Society, 2010, pp. 1–10.
- [32] J.R. Clausen, D.A. Reasor, C.K. Aidun, Parallel performance of a lattice-Boltzmann/finite element cellular blood flow solver on the IBM Blue Gene/P architecture, *Comput. Phys. Commun.* 181 (6) (2010) 1013–1020.
- [33] B.E. Griffith, Immersed boundary model of aortic heart valve dynamics with physiological driving and loading conditions, *Int. J. Numer. Methods Biomed. Eng.* 28 (3) (2012) 317–345.
- [34] S.C. Shadden, M. Astorino, J.-F. Gerbeau, Computational analysis of an aortic valve jet with Lagrangian coherent structures, *Chaos* 20 (1) (2010) 017512.
- [35] S.C. Shadden, A. Arzani, Lagrangian postprocessing of computational hemodynamics, *Ann. Biomed. Eng.* 43 (1) (2015) 41–58.
- [36] D. Rossinelli, Y.-H. Tang, K. Lykov, D. Alexeev, M. Bernaschi, P. Hadjidoukas, M. Bisson, W. Joubert, C. Conti, G. Karniadakis, et al., The in-silico lab-on-a-chip: petascale and high-throughput simulations of microfluidics at cell resolution, in: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ACM, 2015, p. 2.
- [37] P. Perdikaris, L. Grinberg, G.E. Karniadakis, Multiscale modeling and simulation of brain blood flow, *Phys. Fluids (1994–present)* 28 (2) (2016) 021304.
- [38] B.E. Griffith, R.D. Hornung, D.M. McQueen, C.S. Peskin, An adaptive, formally second order accurate version of the immersed boundary method, *J. Comput. Phys.* 223 (1) (2007) 10–49.
- [39] C.A. Taylor, T.J. Hughes, C.K. Zarins, Finite element modeling of blood flow in arteries, *Comput. Methods Appl. Mech. Eng.* 158 (1) (1998) 155–196.
- [40] A. Updegrove, N.M. Wilson, J. Mewkow, H. Lan, A.L. Marsden, S.C. Shadden, SimVascular: an open source pipeline for cardiovascular simulation, *Ann. Biomed. Eng.* (2016) 1–17.
- [41] P. Seil, S. Pirker, LBDEMcoupling: open-source power for fluid-particle systems, in: *Proceedings of the 7th International Conference on Discrete Element Methods*, Springer, 2017, pp. 679–686.
- [42] P. Seil, S. Pirker, T. Lichtenegger, Onset of sediment transport in mono- and bidisperse beds under turbulent shear flow, *Comput. Part. Mech.* (2017) 1–10.
- [43] L. Peng, K.-i. Nomura, T. Oyakawa, R.K. Kalia, A. Nakano, P. Vashishta, Parallel lattice Boltzmann flow simulation on emerging multi-core platforms, in: *European Conference on Parallel Processing*, Springer, 2008, pp. 763–777.
- [44] E.S. Boek, M. Venturoli, Lattice-Boltzmann studies of fluid flow in porous media with realistic rock geometries, *Comput. Math. Appl.* 59 (7) (2010) 2305–2314.
- [45] M.A. Spaid, F.R. Phelan Jr., Lattice Boltzmann methods for modeling microscale flow in fibrous porous media, *Phys. Fluids* 9 (9) (1997) 2468–2474.
- [46] X. Yu, J. Tan, S. Diamond, Hemodynamic force triggers rapid NETosis within sterile thrombotic occlusions, *J. Thromb. Haemost.* 16 (2) (2018) 316–329.
- [47] X. He, S. Chen, R. Zhang, A lattice Boltzmann scheme for incompressible multiphase flow and its application in simulation of Rayleigh–Taylor instability, *J. Comput. Phys.* 152 (2) (1999) 642–663.
- [48] X. Shan, H. Chen, Lattice Boltzmann model for simulating flows with multiple phases and components, *Phys. Rev. E* 47 (3) (1993) 1815.
- [49] D.A. Reasor, J.R. Clausen, C.K. Aidun, Coupling the lattice-Boltzmann and spectrin-link methods for the direct numerical simulation of cellular blood flow, *Int. J. Numer. Methods Fluids* 68 (6) (2012) 767–781.
- [50] G. Závodszy, B. van Rooij, V. Azizi, A. Hoekstra, Cellular level in-silico modeling of blood rheology with an improved material model for red blood cells, *Front. Physiol.* 8 (2017) 563.
- [51] T. Hyakutake, S. Nagai, Numerical simulation of red blood cell distributions in three-dimensional microvascular bifurcations, *Microvasc. Res.* 97 (2015) 115–123.
- [52] X. Shi, S. Wang, S. Zhang, Numerical simulation of the transient shape of the red blood cell in microcapillary flow, *J. Fluids Struct.* 36 (2013) 174–183.
- [53] J. Latt, *Hydrodynamic Limit of Lattice Boltzmann Equations* (Ph.D. thesis), University of Geneva, 2007.
- [54] J. Latt, Palabos, Parallel Lattice Boltzmann Solver, 2009.
- [55] S. Plimpton, Fast parallel algorithms for short-range molecular dynamics, *J. Comput. Phys.* 117 (1) (1995) 1–19.
- [56] S.T. Ollila, C. Denniston, M. Karttunen, T. Ala-Nissila, Fluctuating lattice-Boltzmann model for complex fluids, *J. Chem. Phys.* 134 (6) (2011) 064902.
- [57] F. Mackay, S.T. Ollila, C. Denniston, Hydrodynamic forces implemented into LAMMPS through a lattice-Boltzmann fluid, *Comput. Phys. Commun.* 184 (8) (2013) 2021–2031.
- [58] J. Zhang, P.C. Johnson, A.S. Popel, Red blood cell aggregation and dissociation in shear flows simulated by lattice Boltzmann method, *J. Biomech.* 41 (1) (2008) 47–55.
- [59] T. Krüger, F. Varnik, 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 (12) (2011) 3485–3505.
- [60] R.M. MacMeccan, J. Clausen, G. Neitzel, C. Aidun, Simulating deformable particle suspensions using a coupled lattice-Boltzmann and finite-element method, *J. Fluid Mech.* 618 (2009) 13–39.
- [61] J. Tan, W. Keller, S. Sohrabi, J. Yang, Y. Liu, Characterization of nanoparticle dispersion in red blood cell suspension by the lattice Boltzmann-immersed boundary method, *Nanomaterials* 6 (2) (2016) 30.
- [62] S. Chen, G.D. Doolen, Lattice Boltzmann method for fluid flows, *Annu. Rev. Fluid Mech.* 30 (1) (1998) 329–364.
- [63] X. He, L.-S. Luo, Lattice Boltzmann model for the incompressible Navier–Stokes equation, *J. Stat. Phys.* 88 (3–4) (1997) 927–944.
- [64] S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*, Clarendon, 2001.
- [65] A.J. Ladd, Numerical simulations of particulate suspensions via a discretized Boltzmann equation. Part 1. Theoretical foundation, *J. Fluid Mech.* 271 (1994) 285–309.
- [66] T. Inamuro, M. Yoshino, F. Ogino, Accuracy of the lattice Boltzmann method for small Knudsen number with finite Reynolds number, *Phys. Fluids (1994–present)* 9 (11) (1997) 3535–3542.
- [67] Y. Qian, D. d’Humières, P. Lallemand, Lattice BGK models for Navier–Stokes equation, *EPL (Europhys. Lett.)* 17 (6) (1992) 479.
- [68] Z. Guo, C. Zheng, B. Shi, Discrete lattice effects on the forcing term in the lattice Boltzmann method, *Phys. Rev. E* 65 (4) (2002) 046308.
- [69] M.L. Parks, R.B. Lehoucq, S.J. Plimpton, S.A. Silling, Implementing peridynamics within a molecular dynamics code, *Comput. Phys. Commun.* 179 (11) (2008) 777–783.
- [70] S.A. Silling, Reformulation of elasticity theory for discontinuities and long-range forces, *J. Mech. Phys. Solids* 48 (1) (2000) 175–209.
- [71] J. Monaghan, R. Gingold, Shock simulation by the particle method SPH, *J. Comput. Phys.* 52 (2) (1983) 374–389.
- [72] J.J. Monaghan, Smoothed particle hydrodynamics, *Annu. Rev. Astron. Astrophys.* 30 (1992) 543–574.
- [73] R.D. Groot, P.B. Warren, et al., Dissipative particle dynamics: bridging the gap between atomistic and mesoscopic simulation, *J. Chem. Phys.* 107 (11) (1997) 4423.
- [74] Y. Afshar, F. Schmid, A. Pishevar, S. Worley, Exploiting seeding of random number generators for efficient domain decomposition parallelization of dissipative particle dynamics, *Comput. Phys. Commun.* 184 (4) (2013) 1119–1128.
- [75] T. Ihle, D. Kroll, Stochastic rotation dynamics: a Galilean-invariant mesoscopic model for fluid flow, *Phys. Rev. E* 63 (2) (2001) 020201.
- [76] S. Genheden, J.W. Essex, A simple and transferable all-atom/coarse-grained hybrid model to study membrane processes, *J. Chem. Theory Comput.* 11 (10) (2015) 4749–4759.
- [77] D.A. Fedosov, B. Caswell, G.E. Karniadakis, A multiscale red blood cell model with accurate mechanics, rheology, and dynamics, *Biophys. J.* 98 (10) (2010) 2215–2225.
- [78] I.V. Pivkin, G.E. Karniadakis, Accurate coarse-grained modeling of red blood cells, *Phys. Rev. Lett.* 101 (11) (2008) 118105.
- [79] M. Dao, J. Li, S. Suresh, Molecularly based analysis of deformation of spectrin network and human erythrocyte, *Mater. Sci. Eng. C* 26 (8) (2006) 1232–1244.
- [80] J. Tan, S. Sohrabi, R. He, Y. Liu, Numerical simulation of cell squeezing through a micropore by the immersed boundary method, *Proc. Inst. Mech. Eng. Part C: J. Mech. Eng. Sci.* (2017), 0954406217730850.
- [81] D.A. Reasor Jr., M. Mehrabadi, D.N. Ku, C.K. Aidun, Determination of critical parameters in platelet margination, *Ann. Biomed. Eng.* 41 (2) (2013) 238–249.
- [82] J.B. Freund, M. Orescanin, Cellular flow in a small blood vessel, *J. Fluid Mech.* 671 (2011) 466–490.
- [83] Z.-G. Feng, E.E. Michaelides, The immersed boundary-lattice Boltzmann method for solving fluid–particles interaction problems, *J. Comput. Phys.* 195 (2) (2004) 602–628.
- [84] Y. Liu, Y. Mori, Properties of discrete delta functions and local convergence of the immersed boundary method, *SIAM J. Numer. Anal.* 50 (6) (2012) 2986–3015.
- [85] X. Yang, X. Zhang, Z. Li, G.-W. He, A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations, *J. Comput. Phys.* 228 (20) (2009) 7821–7836.
- [86] D. Goldstein, R. Handler, L. Sirovich, Modeling a no-slip flow boundary with an external force field, *J. Comput. Phys.* 105 (2) (1993) 354–366.
- [87] J. Wu, C.K. Aidun, Simulating 3D deformable particle suspensions using lattice Boltzmann method with discrete external boundary force, *Int. J. Numer. Methods Fluids* 62 (7) (2010) 765–783.
- [88] M.-C. Lai, C.S. Peskin, An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, *J. Comput. Phys.* 160 (2) (2000) 705–719.
- [89] D.J. Mavriplis, A. Jameson, Multigrid solution of the Navier–Stokes equations on triangular meshes, *AIAA J.* 28 (8) (1990) 1415–1425.



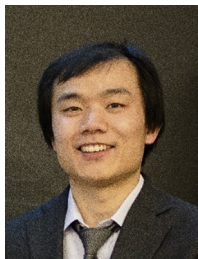
- [90] T.J. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, *Comput. Methods Appl. Mech. Eng.* 194 (39) (2005) 4135–4195.
- [91] B.E. Griffith, R.D. Hornung, D.M. McQueen, C.S. Peskin, Parallel and adaptive simulation of cardiac fluid dynamics, in: *Advanced Computational Infrastructures for Parallel and Distributed Adaptive Applications*, 2010, pp. 105.
- [92] A.J. Ladd, Sedimentation of homogeneous suspensions of non-Brownian spheres, *Phys. Fluids* (1994–present) 9 (3) (1997) 491–499.
- [93] A. Ladd, R. Verberg, Lattice-Boltzmann simulations of particle–fluid suspensions, *J. Stat. Phys.* 104 (5–6) (2001) 1191–1251.
- [94] G.B. Jeffery, The motion of ellipsoidal particles immersed in a viscous fluid, *Proc. R. Soc. Lond. A: Math. Phys. Eng. Sci.* 102 (1922) 161–179.
- [95] R. Fåhræus, T. Lindqvist, The viscosity of the blood in narrow capillary tubes, *Am. J. Physiol. Leg. Content* 96 (3) (1931) 562–568.
- [96] A. Pries, D. Neuhaus, P. Gaehgtgens, Blood viscosity in tube flow: dependence on diameter and hematocrit, *Am. J. Physiol. Heart Circ. Physiol.* 263 (6) (1992) H1770–H1778.
- [97] J.B. Freund, Numerical simulation of flowing blood cells, *Annu. Rev. Fluid Mech.* 46 (2014) 67–95.
- [98] L. Mountrakis, E. Lorenz, O. Malaspinas, S. Alowayyed, B. Chopard, A.G. Hoekstra, Parallel performance of an IB-LBM suspension simulation framework, *J. Comput. Sci.* 9 (2015) 45–50.
- [99] K. Müller, D.A. Fedosov, G. Gompper, Margination of micro- and nano-particles in blood flow and its effect on drug delivery, *Sci. Rep.* 4 (2014).
- [100] Y. Geng, P. Dalhaimer, S. Cai, R. Tsai, M. Tewari, T. Minko, D.E. Discher, Shape effects of filaments versus spherical particles in flow and drug delivery, *Nat. Nanotechnol.* 2 (4) (2007) 249–255.
- [101] J. Boyle, Microcirculatory hematocrit and blood flow, *J. Theor. Biol.* 131 (2) (1988) 223–229.
- [102] J. Baschnagel, H. Meyer, J. Wittmer, I. Kulić, H. Mohrbach, F. Ziebert, G.-M. Nam, N.-K. Lee, A. Johnner, Semiflexible chains at surfaces: worm-like chains and beyond, *Polymers* 8 (8) (2016) 286.
- [103] C. Ness, V.V. Palyulin, R. Milkus, R. Elder, T. Sirk, A. Zaccone, Nonmonotonic dependence of polymer–glass mechanical response on chain bending stiffness, *Phys. Rev. E* 96 (3) (2017) 030501.
- [104] K. Vahidkhah, S.L. Diamond, P. Bagchi, Platelet dynamics in three-dimensional simulation of whole blood, *Biophys. J.* 106 (11) (2014) 2529–2540.
- [105] L.M. Crowl, A.L. Fogelson, Computational model of whole blood exhibiting lateral platelet motion induced by red blood cells, *Int. J. Numer. Methods Biomed. Eng.* 26 (3–4) (2010) 471–487.
- [106] M. Saadatmand, T. Ishikawa, N. Matsuki, M. Jafar Abdekhodaie, Y. Imai, H. Ueno, T. Yamaguchi, Fluid particle diffusion through high-hematocrit blood flow within a capillary tube, *J. Biomech.* 44 (1) (2011) 170–175.



**Talid R. Sinno** research program is broadly aimed at the theoretical and computational study of nano and microstructural evolution, particularly nucleation and growth, in condensed matter. He is currently pursuing microdefect formation in crystalline semiconductors, colloidal crystallization of micron-scale particles with tunable interactions, and platelet aggregation in blood. Talid's lab has developed an extensive suite of simulation tools for studying these phenomena, including large-scale molecular dynamics, multiscale kinetic Monte Carlo, and continuum models.



**Scott L. Diamond**, Ph.D. (B.S., Cornell University; Ph.D., Rice University), is the founding director of the Penn Center for Molecular Discovery. He is the Arthur E. Humphrey Professor of Chemical and Biomolecular Engineering. Dr. Diamond researches biotechnologies in several key areas: endothelial mechanobiology, blood clot dissolving therapies, blood systems biology, nonviral gene therapy, and high throughput drug discovery. He has produced over 180 publications and patents and has served on advisory committees to NSF, NIH, AHA, and NASA, and has consulted extensively for industry and government. Diamond is the recipient of the NSF National Young Investigator Award, the NIH FIRST Award, the American Heart Association Established Investigator Award, the AIChE Allan P. Colburn Award, and the George Heilmeyer Excellence in Research Award. Dr. Diamond is an elected Fellow of the Biomedical Engineering Society (BMES). Currently, Dr. Diamond is the Director of the Penn Biotechnology Masters Program, one of the largest in the country with more than 130 students. Dr. Diamond also serves as Associate Director of the Institute for Medicine and Engineering (IME).



**Dr. Jifu Tan** is an assistant professor in mechanical engineering at Northern Illinois University. He obtained his PhD from Lehigh University. He was then a postdoc at the University of Pennsylvania. His research focuses on computational modeling of biological flows.