Contents lists available at ScienceDirect

Computer Physics Communications

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

OpenFSI: A highly efficient and portable fluid–structure simulation package based on immersed-boundary method[☆]

Huilin Ye[a], Zhiqiang Shen[a], Weikang Xian[a], Teng Zhang[b], Shan Tang[c,*], Ying Li[a,*]

[a] Department of Mechanical Engineering and Polymer Program, Institute of Materials Science, University of Connecticut, Storrs, CT 06269, USA
[b] Department of Mechanical and Aerospace Engineering, Syracuse University, Syracuse, NY 13244, USA
[c] State Key Laboratory of Structural Analysis for Industrial Equipment, Department of Engineering Mechanics, Dalian University of Technology, Dalian, 116023, PR China

ARTICLE INFO

Article history:
Received 4 January 2020
Received in revised form 15 June 2020
Accepted 22 June 2020
Available online 30 June 2020

Keywords:
Fluid–structure interaction
Lattice model
Lattice Boltzmann method
Immersed boundary method

ABSTRACT

We have developed a highly efficient and portable fluid–structure interaction (FSI) simulation package, so-called OpenFSI. Within this package, the structure dynamics is accounted by a lattice model (LM) implemented in the framework of Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS), demonstrating the same accuracy as finite element analysis. The fluid flow is resolved by Palabos, which adopts the Lattice Boltzmann method (LBM) to efficiently solve the Boltzmann equation that can recover the Navier–Stokes equation in mesoscale. Additionally, the immersed boundary method (IBM) is employed to couple LM and LBM together, therefore endowing the flexibility to choose alternative solid and fluid solvers. The whole simulation is fulfilled within the framework of Palabos, and the LAMMPS framework is called in Palabos as an external library and coupled through IBM. To demonstrate the capability and accuracy of the proposed package, the validations for the LM are first performed by conducting the deflections of two-dimensional (2D) and three-dimensional (3D) beams in LAMMPS, and comparing the results with those in finite element analysis. Followed are the classical benchmarks of flow passing 2D flexible beam behind a cylinder and 3D flow passing a fixed cylinder. In the results, the free-falling of spheres and flapping of a deformable plate in cross-flow are investigated. Furthermore, the possibility to study complex FSI phenomena is demonstrated by the cases of spheres passing a dam and swimming of microswimmers. Lastly, the efficiency of this simulation package is explored by examining an extremely large system with thousands of red blood cells in blood flow. The OpenFSI package is found to have excellent linear scalability up to 8192 processors, due to the particle-based LM and LBM for structure and fluid flow respectively, as well as advanced cyberinfrastructure of LAMMPS package. Therefore, OpenFSI presents an alternative option to efficiently solve large scale FSI problems, hence to facilitate the unveiling of underlying physical mechanisms.

1. Introduction

Fluid–structure interaction (FSI), a complicated system involving fluid, solid and interaction between each other, has achieved great progress in simulation studies with the advancement of modern computing technology. However, it remains a challenge to fully understand FSI problems with broad applications, due to their multiphysics and multidisciplinary nature [1–6]. The separated fluid and solid have their own governing equations that two systems are required to describe and track them either simultaneously or asynchronously, depending on interaction manner. Moreover, the solid or fluid may partially be controlled or influenced by other physical processes such as electrical, chemical or biological aspect. Therefore, to interpret the motion of the whole system, it requires the synthesis of broad perspectives through connecting different knowledges and skills. Nevertheless, researchers make great efforts and attain increasing advancements in a broad range of FSI applications, including: powder bed fusion in additive manufacturing [7,8]; vascular FSI with red blood cells in microvasculature [9–11]; flapping of creatures like fish swimming and bird flying and nonlivings like flag and foil in the flow [12–14]; transport of contaminants, sediment, or other solutes in rivers and streams [15,16]; turbulent flows that contain particles occurring in a variety of industrial, biological, and environmental processes [17,18]. Despite the theoretical and experimental progress, developing reliable and efficient modeling techniques for FSI problems emerges as an indispensable tool to unveil underlying physical mechanisms of these phenomena.

[☆] The review of this paper was arranged by Prof. David W.Walker.
[*] Corresponding authors.
E-mail addresses: shantang@dlut.edu.cn (S. Tang), yingli@engr.uconn.edu (Y. Li).

https://doi.org/10.1016/j.cpc.2020.107463
0010-4655/© 2020 Elsevier B.V. All rights reserved.

Conventional numerical approaches for FSI problems are usually established based on conforming meshes. Among those, the Arbitrary Lagrangian–Eulerian (ALE) [19–22] and the space–time finite-element method [23,24], as representations of the conforming approaches, are the most well-known methods using boundary-fitted mesh. In these methods, the structure and flow are meshed separately. A Lagrangian mesh is adopted to describe the material points and then capture the motion of the structure. And another one is used to discretize the flow which is required to conform the instantaneous shape of the structure. Using the body-conformal mesh, the boundary condition of the flow is allowed to directly apply on the solid surface. More importantly, the boundary layer can be well resolved through locally refining the flow mesh near the solid surface. However, these advantages are paid by computational cost. The moving of mesh points in bulk flow is needed to follow the boundary of the structure. This may cause the distortion of fluid mesh, especially when encountering large translations or rotations of the structure. Hence, a re-meshing or mesh-updating process is necessary to ensure the conforming of fluid and structure [25]. This process on one hand, consumes large computational time, on the other hand, lowers the accuracy due to the transferring of information from the degenerated mesh to the new one.

Contrary to the conforming mesh-based approaches, the non-body-conformal mesh-based approach is designed to consider fluid and structure as two separated computational fields. Usually, the fluid model is constructed on fixed Cartesian mesh. And a Lagrangian mesh is adopted to track the structure and allowed to move on the top of the background fixed Cartesian mesh. The numerical algorithms to solve fluid and structure motion can perform either sequentially (staggered) or simultaneously (monolithic). Although monolithic techniques demonstrate high numerical convergence and robustness due to the fully-coupled fashion, the price needed to pay is that the monolithic approach requires the writing of a fully-integrated fluid–structure solver, which closes the door on the use of existing fluid and structure software [26,27]. However, the staggered approach outperforms the monolithic approach in terms of the efficiency, and the flexibility of the adoption of existing well-validated fluid and solid solvers, due to the uncoupled fashion. One of the popular staggered approach is so-called immersed boundary method (IBM), which is first proposed by Peskin [28] in 1970s to study the heart valve flow, and then refined by many other researchers with extensive applications in FSI problems [29–40]. In IBM, the interaction is accomplished by the interpolation of interfacial forces and velocities through interpolation functions between fluid and structure domains. The construction of interpolation function should meet the continuity of velocity and force across the fluid–structure interface [41]. Up to now, there are two well-known interpolation functions: reproducing kernel function [42,43] and Dirac delta function [41]. Reproducing kernel function is first proposed in reproducing kernel particle method (RKPM) that is one of the mesh-free methods [42,43]. It is well-known due to the high order of interpolation and the suitability for both uniform and non-uniform meshes. However, a search algorithm should be completed in each time step in RKPM to identify new sets of neighbors after immersed solid advances to a new position, which will consume a lot of computational time. On the contrary, Dirac delta function is constructed based on uniform Cartesian mesh, which acts as a template that is independent on spatial discretization. Therefore, delta function can be easily implemented into an existing fluid or solid solver with high efficiency. One should note that the simplicity of Dirac delta function leads to the drawback that the boundary interface between fluid and solid is smoothed. The boundary condition is applied on the region around the boundary rather than at the actual location. Nevertheless, in terms of efficiency that is essential in modeling complex and large scale FSI problems, Dirac delta function will outperform the reproducing kernel function.

In the meantime, owing to the simplified mesh generation in IBM, the computation on structured meshes becomes more efficient and the domain-decomposition based partition is easier. Most importantly, IBM enables us to have large freedom to choose the solvers for fluid and structure in FSI modeling, thus allowing us to integrate existing packages or software with 'legacy' codes to account for fluid or structure. Since the algorithms used in these packages and software have demonstrated efficiency and accuracy with a variety of benchmarks, to directly utilize them can reduce the time of code development and we just need to focus on the efficiency of the implementation of the interface between these individual packages or software. Among the fluid solvers in computational fluid dynamics, Lattice Boltzmann method (LBM)-based solvers are chosen due to the high parallelization across multiple processors in high-performance-computing (HPC) [44–46]. LBM is a mesoscopic approach to solve the Boltzmann equation that can recover the Navier–Stokes equation [47]. It was first introduced by Feng and Michaelides [48] to fluid–particle interactions and then has been extensively adopted in IBM for a broad range of FSI problems [49–51,35,38,33]. However, none of these are open-source yet. Here, we adopt Palabos to account for fluid dynamics, which is an open-source LBM solver. Most importantly, the LBM in Palabos is implemented using C++ with parallel features through Message Passing Interface (MPI) [52] that it possesses high efficiency when running in the contemporary HPC system. Thanks to this, Palabos has been widely used in the modelings of multiphase flow in porous media [53], turbulent flow [54], biological blood flow [55], to name a few. One of the applications in large scale problems is conducted by Tan et al. [55]. In [55], Palabos is utilized to model the blood flow coupled with the solid solver through IBM. Besides the efficiency of Palabos, to implement solid solver in LAMMPS software is essential to account for the large numbers of red blood cells in blood flow. LAMMPS software is well-known in molecular simulation due to the high efficiency in handling the exascale particle system. It is supported by large on-line communities and has active mailing list. Further, it is easy for user and developer to extend new functionality and features like interaction potentials and fluid models. Therefore, it is natural to build the structure solver on this platform with the destination of developing robust, portable and open-source FSI package.

Among the solid solvers, finite element analysis (FEA) is usually employed in FSI to couple with LBM [30,31,56,35]. Although it demonstrates extensive applications in modeling of large deformation of nonlinear structures [57–59] and high Reynolds number flow [60,56], FEA is not suitable to be implemented within the LAMMPS framework, which favors the particle-based model. On the contrary, Lattice Model (LM) is intrinsically a generalized coarse-grained model, which can be seamlessly integrated into LAMMPS due to its particle-based nature. In particular, LM stands out due to its high efficiency in modeling solid structures [61–63]. It simulates the deformation of solids with a system of discrete particles connected with springs. It has been widely used in rocks [64], concretes [65], and composites [66]. Recently, it was employed to describe non-uniform deformation of soft materials, and to study the dynamics of soft materials in the fluid flow [67,68]. With the same accuracy as FEA, LM can be considered as an alternative approach to study FSI problems. However, most of LMs are limited to the regular lattices, which may prevent the application of LM in FSI. Current study adopts the LM recently derived by Zhang [69], in which the irregular lattice can also demonstrate the same accuracy with FEA for neo-Hookean solids. In this kind of LM, the lattice spring deformation energies are assumed to link with the strain energy in FEA. The spring stiffnesses are obtained by the FEA shape functions directly and the averaged volumetric strain is adopted when calculating the bulk deformation energy.

With the help of IBM, the Palabos and LAMMPS are coupled together to account for fluid and structure dynamics, respectively. As both Palabos and LAMMPS are efficiently parallelized, the coupling plays a key role in determining the efficiency of FSI package. Here, the delta function is adopted in IBM instead of reproducing kernel function which requires a time-consuming searching process. And only regular computation domains are studied in current work that delta function can easily account for. In present work, through performing several classical FSI problems, efficiency and robustness are demonstrated with the proposed framework. We name this simulation package as OpenFSI. Currently, OpenFSI is open-source (https://github.com/huilinye/OpenFSI) with the purpose to help interested users utilize this package to solve their own physical problems.

This paper is organized as follows. In Section 2, the whole computation framework of OpenFSI is given. The LBM is firstly introduced for fluid flow. Then 1D, 2D and 3D LMs are described. And the IBM is discussed in detail to couple the fluid dynamics and immersed structures, especially in terms of the consistent CPU mapping technique in the coupling between individual solvers. In Section 3, numerical validations for the proposed FSI scheme are presented, including deflections of flexible 2D and 3D beams, flow passing a 2D flexible beam behind a cylinder, flow passing a fixed 3D cylinder and deformation of capsule in bi-shear flow. Further studies of complex FSI problems such as flapping of flexible plate in cross-flow and swimming of microswimmers, are examined in Section 4. In addition, a special Section 5 is presented to study the computational efficiency and linear scalability of OpenFSI. Conclusion remarks are provided in Section 6.

2. Computational method

2.1. Overview of the computational framework for OpenFSI

Prior to introduce the detailed numerical methods, we present the whole computational framework of OpenFSI in Fig. 1. A FSI simulation package typically contains three parts: fluid solver, solid solver, and fluid–structure coupling interface. Here the fluid is considered as Newtonian fluid and the flow is incompressible, governed by the Navier–Stokes equations (NSE). We adopt the Palabos that is Lattice Boltzmann method (LBM)-based approach to solve Boltzmann equation that can recover the NSE. Three different flow conditions such as simple shear flow, Poiseuille flow, and uniform flow through configuring different boundary conditions are investigated. The solid structure represented by LM is solved within LAMMPS framework. 1D, 2D, and 3D LMs are implemented by different force fields, which are energetically equivalent to the corresponding continuum models. The force and velocity information on the interfacial region between fluid and structure are transferred upon the IBM. With the force spread from the solid structure, we can advance the LBM and obtain the new velocities in the fluid domain. The fluid velocity information is passed to LAMMPS for structural dynamics, and the structure motion is solved based on the LM. This sustained cycling fulfills the whole FSI process. In the following, we will introduce these solvers one by one.

2.2. Fluid solver: Lattice Boltzmann method (LBM)

Here, the fluid is considered as Newtonian fluid, and the flow model is 2D or 3D incompressible flow. The dynamics of the fluid flow is governed by the NSE, which can be described in an Eulerian coordinate system as:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\frac{1}{\rho^f}\nabla p + \frac{\mu}{\rho^f}\nabla^2\mathbf{u} + \mathbf{F}, \tag{1}$$

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

where ρ^f, $\mathbf{u}$, p are the fluid density, velocity, and pressure, respectively. μ is the dynamic viscosity of the fluid, and $\mathbf{F}$ is the body force.

Owing to its efficiency, in computational fluid dynamics, LBM has been extensively adopted to solve the discrete Boltzmann equation that can recover incompressible NSE through Chapman–Enskog analysis [47], instead of directly solving the NSE. The underlying theory and accuracy can be found in the literature [49, 47]. For sake of simplicity, we briefly summarize the fundamentals of LBM and introduce the model setups in present work. The explicit parameter underpinning LBM is density distribution function $f_i(\mathbf{x}, t)$. It is constructed in the phase space $(\mathbf{x}, \mathbf{e}_i)$, where $\mathbf{x}$ is the position and $\mathbf{e}_i$ is lattice velocity in ith direction. The advance of density distribution function includes two processes: streaming and collision. The linearized Boltzmann equation has the form:

$$(\partial_t + e_{i,}\partial_\alpha)f_i = -\frac{1}{\tau}(f_i - f_i^{eq}) + F_i. \tag{3}$$

The L.H.S of Eq. (3) can be discretized into $f_i(\mathbf{x}+\mathbf{e}_i, t+1) - f_i(\mathbf{x}, t)$, corresponding to the streaming process. The term $-\frac{1}{\tau}(f_i - f_i^{eq})$ in the R.H.S represents the collision process. Here, the collision model is the simplest Bhatnagar–Gross–Krook (BGK) scheme [50]. F_i is a discretized external forcing term.

In our simulation scheme, D3Q19 model is used [70], where the fluid particles have possible discrete velocities stated as Eq. (4) in Box I.

The equilibrium distribution function $f_i^{eq}(\mathbf{x}, t)$ can be calculated as:

$$f_i^{eq}(\mathbf{x}, t) = \omega_i\rho\left[1 + \frac{\mathbf{e}_i\cdot\mathbf{u}}{c_s^2} + \frac{(\mathbf{e}_i\cdot\mathbf{u})^2}{2c_s^4} - \frac{(\mathbf{u})^2}{2c_s^2}\right], \tag{5}$$

where the weighting coefficients $\omega_i = 1/3\ (i = 0)$, $\omega_i = 1/18\ (i = 1-6)$, $\omega_i = 1/36\ (i = 7-18)$. The term c_s represents the sound speed which equals to $\Delta x/(\sqrt{3}\Delta t)$. Δx and Δt are spatial and temporal discretization sizes, respectively. The relaxation time τ is related to the kinematic viscosity in NSE with the form of

$$\nu = (\tau - \frac{1}{2})c_s^2\Delta t. \tag{6}$$

The external forcing term can be discretized by the form [71]:

$$F_i = (1 - \frac{1}{2\tau})\omega_i\left[\frac{\mathbf{e}_i - \mathbf{u}}{c_s^2} + \frac{(\mathbf{e}_i\cdot\mathbf{u})}{c_s^4}\mathbf{e}_i\right]\cdot\mathbf{F}. \tag{7}$$

Once the particle density distributions f_i are known in the whole fluid domain, the fluid density and momentum are calculated as

$$\rho = \sum_i f_i,\quad \rho\mathbf{u} = \sum_i f_i\mathbf{e}_i + \frac{1}{2}\mathbf{F}\Delta t. \tag{8}$$

In the current LBM scheme, the Zou/He boundary condition is adopted to account for the pressure and velocity in the computational boundaries [72]. The fluid domain is assumed to be rectangular and three different flow conditions are examined: simple shear flow, Poiseuille flow, and uniform flow. In the simple shear flow, the upper and lower bounds are applied Dirichlet condition to impose the velocities. The boundaries in flow direction are applied with periodic boundary conditions and boundaries in the other directions are Neumann condition for velocity. In the Poiseuille flow, we apply a body force to the fluid in the

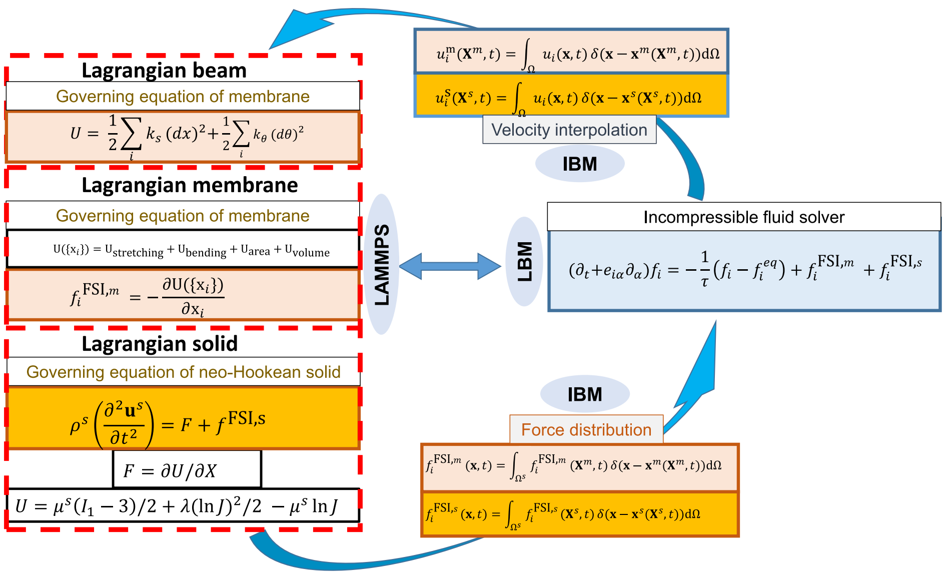

Lagrangian beam
Governing equation of membrane
$U = \frac{1}{2}\sum_i k_s (dx)^2 + \frac{1}{2}\sum_i k_\theta (d\theta)^2$
Lagrangian membrane
Governing equation of membrane
$U(\{x_i\}) = U_{stretching} + U_{bending} + U_{area} + U_{volume}$
$f_i^{FSI,m} = -\frac{\partial U(\{x_i\})}{\partial x_i}$
Lagrangian solid
Governing equation of neo-Hookean solid
$\rho^s\left(\frac{\partial^2 \mathbf{u}^s}{\partial t^2}\right) = F + f^{FSI,s}$
$F = \partial U/\partial X$
$U = \mu^s(I_1 - 3)/2 + \lambda(\ln J)^2/2 - \mu^s \ln J$
LAMMPS
LBM
$u_i^m(\mathbf{X}^m, t) = \int_\Omega u_i(\mathbf{x}, t)\,\delta(\mathbf{x} - \mathbf{x}^m(\mathbf{X}^m, t))d\Omega$
$u_i^s(\mathbf{X}^s, t) = \int_\Omega u_i(\mathbf{x}, t)\,\delta(\mathbf{x} - \mathbf{x}^s(\mathbf{X}^s, t))d\Omega$
Velocity interpolation
IBM
Incompressible fluid solver
$(\partial_t + e_{i\alpha}\partial_\alpha)f_i = -\frac{1}{\tau}(f_i - f_i^{eq}) + f_i^{FSI,m} + f_i^{FSI,s}$
IBM
Force distribution
$f_i^{FSI,m}(\mathbf{x}, t) = \int_{\Omega^s} f_i^{FSI,m}(\mathbf{X}^m, t)\,\delta(\mathbf{x} - \mathbf{x}^m(\mathbf{X}^m, t))d\Omega$
$f_i^{FSI,s}(\mathbf{x}, t) = \int_{\Omega^s} f_i^{FSI,s}(\mathbf{X}^s, t)\,\delta(\mathbf{x} - \mathbf{x}^s(\mathbf{X}^s, t))d\Omega$

Fig. 1. Schematic of the computational framework of OpenFSI.

$$[\mathbf{e}_0, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_5, \mathbf{e}_6, \mathbf{e}_7, \mathbf{e}_8, \mathbf{e}_9, \mathbf{e}_{10}, \mathbf{e}_{11}, \mathbf{e}_{12}, \mathbf{e}_{13}, \mathbf{e}_{14}, \mathbf{e}_{15}, \mathbf{e}_{16}, \mathbf{e}_{17}, \mathbf{e}_{18}] = \begin{bmatrix} 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}. \tag{4}$$

Box I.

whole domain that mimics the pressure-driven profile. Therefore, a periodic boundary condition is applied in the flow direction and other directions are imposed on wall condition. In Palabos, the wall condition is fulfilled by the bounce-back scheme [47]. In the uniform flow, the uniform velocity boundary is exerted on the inlet and Neumman condition for velocity in the outlet. The other directions are applied with Neumman condition for velocity and Dirichlet condition for pressure. Besides the boundary condition, the unit conversion is internally implemented in the Palabos. The units are denoted as lattice units. To indicate a dimensional physical quantity, a reference scale is required. For example, the length reference in Palabos is usually the length of the resolution of fluid mesh Δx, therefore, for a quantity with length l will be denoted as $l^* = l/\Delta x$ in lattice unit. Also, the time step Δt is typically chosen as the time reference. In mechanics, the dimension of any quantity can be expressed as a combination of length, time, and mass. The mass reference in Palabos can be chosen as the fluid density $\Delta\rho$ that is a combination of the dimensions of length and mass. Then any mechanical quantity p has a reference C_p in Palabos, where C_p is defined as $\Delta x^m \Delta t^n \rho^q$. The superscripts m, n, and q are determined by the dimension of quantity p. Further detailed information of unit conversion can be found in [73].

2.3. Lattice model (LM) for immersed structures

Fig. 2 presents the relationship between the LM and continuum model. LM should intrinsically reflect the same macroscopic properties of the continuum model, such as the in-plane shear, out-of-plane bending, and Young's modulus, etc. The typical parameters in LM are spring constant (linear and angle springs) and some constraints (area and volume). Before solving realistic problems, the parameters in the LM should be first calibrated to reproduce the same behaviors of structures in the continuum model. After the relationship between LM parameters and macroscopic properties is established, the LM can be adopted to model solid structures due to its efficiency and accuracy. Next, we will introduce the LM and discuss the relationship between LM parameters and macroscopic properties of the continuum model in 1D, 2D and 3D.

2.3.1. 1D lattice beam model

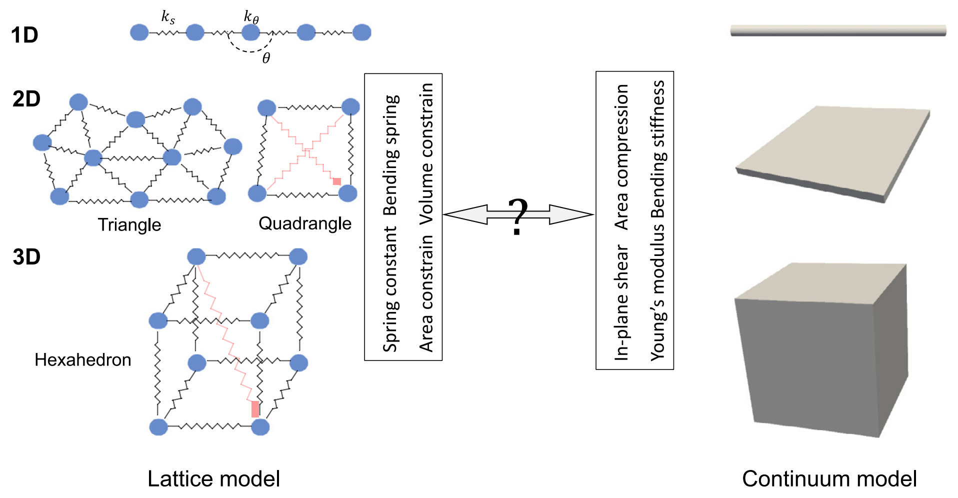
In this work, a 1D coarse-grained lattice model is employed to account for the 1D continuum beam. As shown in Fig. 2, the beam is discretized into particles consecutively connected with linear springs. Additionally, an angle spring is exerted between the adjacent linear springs. For this model, the total energy U_{1D} of the beam has the form:

$$\begin{aligned} U_{1D} &= U_{\text{linear}} + U_{\text{angle}} + U_{\text{torsion}} \\ &= \frac{1}{2}\sum_i k_s(dx)^2 + \frac{1}{2}\sum_j k_\theta(d\theta)^2 + \frac{1}{2}\sum_k k_\tau(d\tau)^2, \end{aligned} \tag{9}$$

where the dx is the stretch of linear spring, $d\theta$ is the angle variance of spring angle, and $d\tau$ is the torsion angle. k_s, k_θ, and k_τ are linear, angle, and torsion spring constants, respectively. From the beam model theory [74–76], we can directly obtain the relationship between lattice force constants and macroscopic properties of the continuum model as follows:

$$k_s = \frac{EA}{r_0}, \quad k_\theta = \frac{EI}{r_0}, \quad k_\tau = \frac{GJ_0}{r_0}, \tag{10}$$

where r_0 is the radius of a beam, A is the area of cross-section of the beam, I and J_0 are the in-plane and out-of-plane moments of inertia, respectively. E and G are Young's and shear moduli of the beam, respectively. It should be noted that here, the torsion



1D
k_s
k_θ
θ
2D
Triangle
Quadrangle
3D
Hexahedron
Spring constant Bending spring
Area constrain Volume constrain
?
In-plane shear Area compression
Young's modulus Bending stiffness
Lattice model
Continuum model

Fig. 2. Sketch of lattice models and continuum models. For sake of clarity, only one diagonal spring (red spring) in 3D hexahedron element is shown. Actually, there are total 28 springs in the 3D hexahedron element, where any two nodes are connected with one spring. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

of the beam is not considered. Because the diameters of the filamentous micro-organisms we study are too small, the torsion has a negligible effect on the dynamics of the beam.

2.3.2. 2D and 3D solid lattice model

The 2D and 3D solid lattice models used in this work are formulated by Zhang [69], and they are adapted to be implemented in LAMMPS accordingly. Here, the main framework and related underlying principles are provided in case one has interests to reproduce the same work and modify them accordingly for specific physical problems. At the same time, more details can be found in Ref. [69]. In the following, neo-Hookean material [77] is adopted to describe the solid part in LMs. It is a simple and widely used constitutive model for hyperelastic materials. The strain energy density U_{neo} of a neo-Hookean solid can be described as [77]:

$$U_{\mathrm{neo}} = \mu^s (I_1 - 3)/2 - \mu^s \ln J + \lambda (\ln J)^2/2, \tag{11}$$

where μ^s is the shear modulus, λ is the Lame constant, I_1 is the first invariant of the right Cauchy–Green deformation tensor, $I_1 = \lambda_1^2 + \lambda_2^2 + 1$ for plane strain deformation and $I_1 = \lambda_1^2 + \lambda_2^2 + \lambda_3^2$ for 3D deformation, $\lambda_i, i = 1, 2, 3$ is the principal stretches. J is the determinant of the deformation gradient tensor $\mathbf{F}_{\mathrm{de}}$.

First, we discuss the 2D square LM. The essential part relies on the estimation of strain energy density based on spring stretch and area change in a square lattice unit (Fig. 2). For example, the energies associated with I_1 and area change J are:

$$U_{I_1} = \frac{1}{2}\mu^s (I_1 - 3) = \frac{\mu^s}{2A_0}[(r_{12}^2 + r_{23}^2 + r_{34}^2 + r_{14}^2)/6 + (r_{13}^2 + r_{24}^2)/3 - 2], \tag{12}$$

$$U_{\mathrm{J}} = \frac{1}{2}\lambda (\ln J)^2 - \mu^s \ln J = \frac{1}{2}\lambda (\ln(A/A_0))^2 - \mu^s \ln(A/A_0), \tag{13}$$

where r_{ij} denotes the length of the spring, $i, j = 1, 2, 3, 4$ are indexes of the nodes in the square lattice. A and A_0 are the areas of the square lattice unit and corresponding initial value, respectively.

Then we demonstrate the equivalence between the LM and FEA based on irregular lattices. In the framework of finite element analysis (FEA), the deformation gradient tensor in an irregular element is [78]:

$$F_{\mathrm{ij}} = \frac{\partial x_i}{\partial X_j} = x_i^a \frac{\partial N^a}{\partial X_j}, \tag{14}$$

where $N^a(X_1, X_2)$ is the shape function, $a = 1, 2, 3, 4$, and $i, j = 1, 2$. In 2D plane strain problems, we have $I_1 = F_{\mathrm{ij}} F_{\mathrm{ij}} + 1$, then the strain energy corresponding to I_1 can be obtained:

$$A_0 U_{I_1} = \int \frac{1}{2}\mu^s (x_i^a x_i^b \frac{\partial N^a}{\partial X_j}\frac{\partial N^b}{\partial X_j} - 2) dA_0 = -\frac{1}{2} k_{\mathrm{ab}} x_i^a x_i^b - \mu^s A_0, \tag{15}$$

where $k_{\mathrm{ab}} = -\int \mu^s \frac{\partial N^a}{\partial X_j}\frac{\partial N^b}{\partial X_j} dA_0$. In FEA, the shape function should meet $\sum_{a=1}^4 N^a = 1$, we have the following relations

$$\sum_{a=1}^{4} k_{\mathrm{ab}} = -\mu^s \int (\sum_{a=1}^{4} \frac{\partial N^a}{\partial X_j}) \frac{\partial N^b}{\partial X_j} dA_0 = 0, \tag{16}$$

We substitute Eq. (16) into Eq. (15), considering the symmetry of k_{ab}, the strain energy can be expressed by summation of energies for springs in the lattice structure,

$$U_{I_1} = \frac{1}{2} A_0^{-1} \sum_{b=2, b>a}^{4} \sum_{a=1}^{3} k_{\mathrm{ab}} r_{\mathrm{ab}}^2 - \mu^s, \tag{17}$$

where k_{ab} is the spring constant. For irregular lattices, k_{ab} can be calculated through Gaussian quadrature with isoparametric mapping. And the energy associated with the area strain is calculated by the same method used in F-bar method [79]. It should be noted that, the above demonstration for the equivalence between LM and FEA is based on the construction of I_1, therefore, this kind of analysis will be suitable for other hyper-elastic materials such as Gent [80] and Arruda–Boyce [81] models, which are formulated based on I_1.

For the 3D problem, the solid structure is partitioned into hexahedron elements, then one hexahedron element has 28 lattice springs. The strain energy associated with I_1 can be written as

$$U_{I_1} = \frac{1}{2} V_0^{-1} \sum_{b=2, b>a}^{8} \sum_{a=1}^{7} k_{\mathrm{ab}} r_{\mathrm{ab}}^2 - \mu^s, \tag{18}$$

and the energy associated with the volumetric deformation J is

$$U_{\mathrm{J}} = \frac{1}{2}\lambda (\ln(V/V_0))^2 - \mu^s \ln(V/V_0), \tag{19}$$

where V and V_0 are the volumes of the hexahedron element and the corresponding initial value, respectively. Similar to the 2D case, the spring constant k_{ab} and volumetric energy can be calculated. The area in 2D case is relatively easy to be obtained, here we introduce the form to estimate the 3D volume of hexahedron element. With the form of deformation gradient tensor $F_{ij} = x_i^a \frac{\partial N^a}{\partial X_j}$, we can have the volume of the hexahedron element:

$$\begin{aligned} V &= \int \det(\mathbf{F}) d\xi = \int \frac{1}{6} \epsilon_{ijk}\epsilon_{lmn} F_{li} F_{mj} F_{nk} d\xi, \\ &= \int \frac{1}{6} \epsilon_{ijk}\epsilon_{lmn} x_l^a \frac{\partial N^a}{\partial \xi_i} x_m^b \frac{\partial N^b}{\partial \xi_j} x_n^c \frac{\partial N^c}{\partial \xi_k}. \end{aligned} \tag{20}$$

Next, we introduce the process about how to pre-calculate the spring constants to prepare for the simulation. First, we construct a model in FEA, and then mesh it with either quadrilateral or hexahedron element. Using this configuration, the values of spring constant can be computed based on Eq. (16). It should be noted that the values of spring constants are only dependent on the initial nodal positions inside an element. Therefore, it is not necessary to repeatedly perform Gaussian quadrature during the simulations. With these spring constants, we can easily calculate the bonded forces in LAMMPS. Along with the result of energy-relevant to the area or volume deformation, we can solve the motion equation of the solid.

2.3.3. Shell and membrane lattice model

The shell and membrane lattice models adopted here are originally developed by Fedosov [82]. The accuracy and efficiency have been further confirmed by our recent work [9]. In the following, we just briefly represent them accordingly for the completeness of the records. Shell and membrane usually point to the open and closed 2D structures in 3D space, respectively. In FSI, they are adopted to describe biological particles such as capsules [83–85] and vesicles [86]. And they can play significant roles in the modeling of red blood cells (RBCs) in blood flow [87,88,40]. The elastic particles are considered as membrane enclosing internal fluid and immersed in the external fluid environment. The internal and external fluid can be either the same (fluid-filled membrane) or not (cytoplasm and plasma fluid). In simulations, the membrane is discretized into many interacting particles, which are usually connected in a triangular pattern (cf. Fig. 2). The mechanical properties of the elastic particle are accounted for by exerting a variety of constraints on the membrane. Potential function used to describe the membrane is given as:

$$U(\{\mathbf{x}_i\}) = U_{stretching} + U_{bending} + U_{area} + U_{volume}, \tag{21}$$

where $U_{stretching}$ represents the in-plane stretching property of membrane. $U_{bending}$ denotes the out-of-plane bending resistance. U_{area} and U_{volume} ensure the total area and volume conservation, which correspond to the area incompressibility of membrane and incompressibility of the internal fluid, respectively. It should be emphasized that the choices of the potential terms will correspond to membrane models. For example, a capsule usually has no resistance to the out-of-plane bending [58], hence, the potential $U_{bending}$ should be removed from the total potential. Here, we use the RBC membrane model as an example to introduce the different potential forms. The stretching potential $U_{stretching}$ consists of two parts: attractive worm-like chain model (WLC) and repulsive power function (POW). They are expressed by:

$$U_{WLC} = \frac{k_{BT} l_m}{4p} \frac{3x^2 - 2x^3}{1 - x}, \; U_{POW} = \frac{k_p}{l}, \tag{22}$$

where k_B is the Boltzmann constant. $x = l/l_m \in (0, 1)$, l is the length of the spring and l_m is the maximum spring extension. p is the persistent length, and k_p is the POW force coefficient. The bending potential has the form

$$U_{bending} = \sum_{k \in 1 \ldots N_s} k_b [1 - \cos(\theta_k - \theta_0)], \tag{23}$$

where k_b is the bending stiffness. θ_k and θ_0 are the dihedral angle between two adjacent triangular elements and its initial value, respectively. N_s denotes the total number of dihedral angles. To ensure the conservation of total area for the RBC, local, and global area constraints are applied. They are expressed as:

$$U_{area} = \sum_{k=1 \ldots N_t} \frac{k_d (A_k - A_{k0})^2}{2A_{k0}} + \frac{k_a (A_t - A_{t0})^2}{2A_t}, \tag{24}$$

where the first term represents the local area constraint, and N_t is the total number of triangular elements. A_k and A_{k0} denote the kth element area and its initial area, respectively. k_d is the corresponding spring constant. The second term is the global area constraint. A_t and A_{t0} are the total area and its initial value, respectively. k_a is the spring constant. The total volume constraint is also imposed by a harmonic potential

$$U_{volume} = \frac{k_v (V - V_0)}{2V_0}, \tag{25}$$

where k_v is the spring constant. V and V_0 are the total volume and its initial value, respectively. With the potential forms, the nodal force exerted on the membrane can be obtained as the derivation of the potential as:

$$\mathbf{f}_i = -\frac{\partial U(\{\mathbf{x}_i\})}{\partial \mathbf{x}_i}. \tag{26}$$

The coefficients in the above potential forms are usually chosen according to the corresponding macroscopic properties such as shear modulus, bending modulus, bulk modulus, etc. The macroscopic properties can be obtained through experiments as *a priori*. Then we can connect the parameters in coarse-grained shell/membrane model with these macroscopic properties. By extending the linear analysis of a two-dimensional sheet of a spring network built with equilateral triangles, the macroscopic properties can be expressed by the coarse-grained parameters as follows [89,90,82]

$$\begin{aligned} \mu^s &= \frac{\sqrt{3} k_B T}{4 p l_m x_0} \left(\frac{x_0}{2(1 - x_0)^3} - \frac{1}{4(1 - x_0)^2} + \frac{1}{4} \right) + \frac{3\sqrt{3} k_p}{4 l_0^3}, \\ K &= 2\mu^s + k_a + k_d, \\ Y &= \frac{4K\mu^s}{K + \mu^s}, \end{aligned} \tag{27}$$

where μ^s is the shear modulus. K represents the area compression modulus and Y denotes the Young's modulus. In addition to the above potentials, corresponding interaction should be employed to avoid the overlapping between RBCs or nodes that pertain to different structures. For example, there are many RBCs in the flow, the nodes in one RBC cannot penetrate into other RBCs. Here, a short-range and pure repulsive Lennard–Jones potential $F_{LJ}(r) = 4\epsilon[(\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^6]$, $r < r_{cut}$ is introduced, ϵ and σ are depth of the well and zero potential distance, respectively. r_{cut} is the cut-off distance that nodes used to discretize the structure can interact with each other.

2.4. Fluid–structure coupling: Immersed boundary method

Recently, Tang et al. [91] have developed the Multiscale Universal Interface (MUI) for facilitating the coupling effort for a wide range of multiscale/multiphysics simulations. In MUI, all the simulation models can be considered as a cloud of data points, each carrying three essential attributes, such as position, type, and value. Note that a continuum object in the FEA model is discretized into a finite number of nodes and meshes. These nodes are considered to be the data points and their connections (meshes) can be temporarily ignored. Then, the coupling interface within MUI can define a generic push method for solvers to

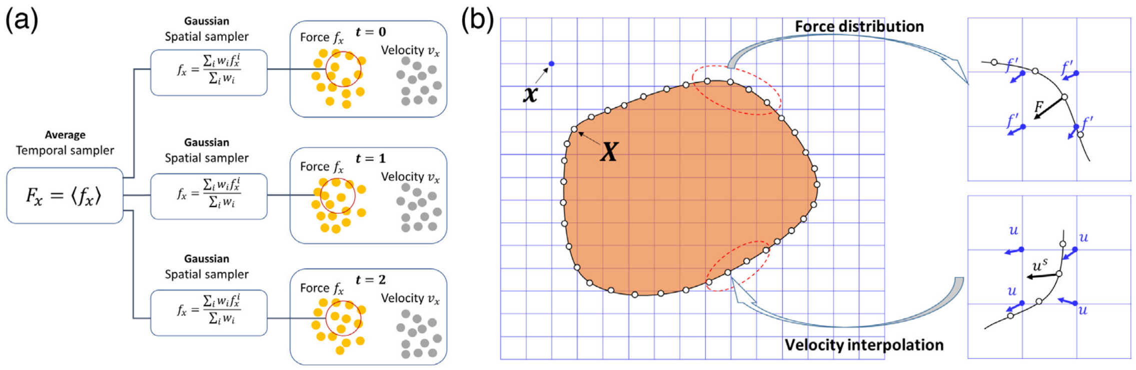


Fig. 3. (a) A data sampler method to interpolate or extrapolate the force and velocity between fluid and solid solvers, adapted from Ref. [91]. (b) Schematic of the IBM. Solid squares represent the Eulerian fluid nodes (x), and solid circles denote vertices of the Lagrangian structure nodes (X).

exchange points carrying different types of data in a unified fashion. For instance, the fluid solver (Palabos) needs the solid force values from solid solver (LAMMPS) at fluid–structure interface; while the solid solver requires the fluid velocity values from a fluid solver at the same boundary. Thus the solvers will take the responsibility to determine which points get pushed in, because the interface region can be tracked during the FSI simulation. It has been confirmed that MUI can provide an efficient and accurate way to couple different solvers together [91,92].

Inspired by the MUI, a generic fetch method for data interpretation on top of data points can be established for interfacial coupling between fluid and solid solvers. However, building such a method is not trivial due to the fact that solvers may be agnostic of the math and method used by their peers. For example, the fluid and solid solvers use the Eulerian and Lagrangian frameworks, respectively. To resolve this issue, a data sampler method, based on the weighted interpolation using nearby points (cf. Fig. 3(a)) [93], can be implemented and used. During this sampling process, the solver will invoke the fetch method with a point of interests and a sampler. Then, the coupling interface will collect all the data points that lie within the sampler's support domain around the point of interest. Afterwards, the coupling interface will feed the sampler with the collected points and let the sampler to perform its own interpolation. Finally, the sampler will return the interpolated result back to the solver through the coupling interface. In this way, the needed value at an arbitrary desired location (e.g. , fluid–structure interface) can be obtained through samplers that interpolate values from nearby points.

Although many sampling methods, such as reproducing kernel particle method [42,94], texture sampling [95], nearest neighbor sampling [96] and moving least-square sampling [97], have achieved great success, for minimally-intrusive FSI coupling, the immersed FSI framework can be considered as one of the most versatile approaches [98,99]. The basic idea for immersed FSI framework is to ensure the non-slip boundary condition on the interface between fluid and solid domains. It, on one hand, lets fluid and structure move with the same velocity, and on the other hand, makes the force acting on the structure also exert on the surrounding fluid (Newton's third law). The IBM is first introduced by Peskin in the 1970s to simulate blood flow around heart valves [28,41]. The detailed mathematical analysis can be found in Ref. [41], here we briefly review the mathematical foundations to explain how to use IBM for coupling fluid flow and immersed structures. As depicted in Fig. 3(b), there exist two coordinate systems, Eulerian and Lagrangian coordinates. The Eulerian variables describe the fluid part and are defined on the fixed Eulerian mesh x that is adopted to solve LBM. The Lagrangian system tracks the immersed structures X (membrane or solid) in the fluid. The Lagrangian quantities are defined in a curvilinear or unstructured mesh which can move on top of the Eulerian mesh. The basic assumption requires the structure moves with the same velocity as the surrounding fluid. However, the Eulerian mesh and Lagrangian mesh are usually not conformed. This leads to the necessity of interpolation between these two systems. The velocity of the structure can be interpolated through surrounding Eulerian fluid mesh velocities. And conversely, the force obtained from the deformed structures should be spread to the nearby Eulerian fluid meshes through interpolation, which will be accepted by LBM as an external force term. The interpolation stencils for the velocity interpolation and force spreading can be constructed according to specific requirements. In the following, we will present the governing equations of IBM.

The fluid domain is represented by Eulerian coordinates $\mathbf{x}$, while the solid structure is represented by Lagrangian coordinates $\mathbf{s}$. Any positions on the structure can be written as $\mathbf{X}(\mathbf{s}, t)$. To satisfy the non-slip boundary condition between structure and fluid domains, the discretized particles should move with the same velocity as the surrounding fluid. That is

$$\frac{\partial \mathbf{X}(\mathbf{s}, t)}{\partial t} = \mathbf{u}(\mathbf{X}(\mathbf{s}, t)). \tag{28}$$

This condition leads to the moving of the immersed structure. The structure force density $\mathbf{F}(\mathbf{s}, t)$ is obtained by the potential functions discussed in above Section 2 due to the motion-induced deformation, and is distributed to the surrounding fluid mesh by

$$\mathbf{f}'(\mathbf{x}, t) = \int_{\Omega^s} \mathbf{F}(\mathbf{X}^s, t)\delta(\mathbf{x} - \mathbf{x}^s(\mathbf{X}^s, t))d\Omega, \tag{29}$$

where δ is a smoothed approximation of the Dirac delta interpolation function. One of the major assumptions for the construction of the Dirac delta function is that the discretized delta function can be factorized,

$$\delta(\mathbf{x} - \mathbf{x}^s(\mathbf{X}^s, t)) = \delta(x - x(\mathbf{X}^s, t))\delta(y - y(\mathbf{X}^s, t))\delta(z - z(\mathbf{X}^s, t)), \tag{30}$$

This ansatz makes the computation simpler, though it is not essential. The assumption lets the interpolation structure become a cubic lattice, and we can construct a stencil in every direction. The accuracy of this scheme depends on the construction of the delta function. In the present study, the so-called 4-points stencil is used and reads

$$\delta(x) = \begin{cases} \frac{1}{8}(3 - 2|x| + \sqrt{1 + 4|x| - 4x^2}), & 0 \le |x| \le 1, \\ \frac{1}{8}(5 - 2|x| + \sqrt{-7 + 12|x| - 4x^2}), & 1 \le |x| \le 2, \\ 0, & 2 \le |x| \end{cases} \tag{31}$$

This stencil provides a support of 64 lattice points. It has been proven to be more stable and exhibits fewer lattice artifacts. The other interpolation stencils can be found in [57,41]. Delta function interpolation is a symmetrical stencil, and it performs well inside the domain. However, when the structure is close to a wall, it will lead to the problem that there are not enough fluid meshes to interpolate. Under this circumstance, a trilinear stencil can be introduced, which only needs 8 points to fulfill the interpolation. One has interests in this is referred to [70].

Then the interpolated FSI force is added back to the LBM solver as a body force and discretized using the form Eq. (7). The same approximation function is necessarily used to obtain the velocities of the Lagrangian structure on the moving boundary. The mathematical form can be written as follows

$$\mathbf{u}^s(\mathbf{X}^s, t) = \int_{\Omega} \mathbf{u}(\mathbf{x}, t)\delta(\mathbf{x} - \mathbf{x}^s(\mathbf{X}^s, t))d\Omega. \tag{32}$$

However, the IBM mentioned above is established on the assumption that the solid has a fiber-like immersed elastic structure. And it requires that the solid should be quite soft and massless. A penalty method is proposed by Kim and Peskin [100] to overcome these defects, which is later further refined by Tian [34]. The penalty scheme (pIBM) has been extensively used to simulate FSI problems [32,38,39]. In the pIBM, the motion of structure is governed by Newton's second law, and the discretized formulation is:

$$m_i \frac{d\mathbf{u}_i^s(\mathbf{X}^s, t)}{dt} = \mathbf{F}_i^{int} + \mathbf{F}_i^{ext}. \tag{33}$$

The internal force $\mathbf{F}_i^{int}$ comes from the internal structure elasticity, and the external force $\mathbf{F}_i^{ext}$ represents the force exerted on the structure by external stimulus, here it points to the fluid surrounding the structure. To advance Eq. (33), we should know the external force $\mathbf{F}_i^{ext}$ first. The penalty scheme to calculate the FSI force is:

$$\mathbf{F}_i^{ext} = \beta[\mathbf{u}^f(t) - \mathbf{u}^s(t)], \tag{34}$$

where $\mathbf{u}^f(t)$ is the fluid velocity at the position where the structure locates, and $\mathbf{u}^s(t)$ is the structure velocity at the same position. β is the penalty parameter. After $\mathbf{F}_i^{ext}$ is known, it can be spread to surrounding fluid meshes by the same scheme Eq. (29). It should be emphasized that the pIBM cannot ensure the non-slip boundary condition. From Eq. (34), we can see only when $\beta \to \infty$, the non-slip boundary condition can be recovered. This is impractical in the numerical simulation due to the stability.

Intrinsically, IBM and pIBM are the same. They only have a difference in the implementation in the numerical scheme. If we let $m_i \to 0$, then Eq. (33) becomes

$$\mathbf{F}_i^{int} = -\mathbf{F}_i^{ext}, \tag{35}$$

which says the FSI force stems from the internal potential of the structure. Furthermore, if we make $\beta \to \infty$, then Eq. (34) can be rewritten as

$$\mathbf{u}^f(t) = \mathbf{u}^s(t), \tag{36}$$

which is the requirement for the non-slip boundary condition in IBM. Although both of these two schemes essentially reflect the interaction, they have different applications of FSI problems. The IBM performs well in the modeling of soft materials such as membranes, capsules and vesicles, because it accurately ensures the non-slip boundary condition. Whereas pIBM outperforms when it occurs to the structure with finite mass and high stiffness. We have implemented both IBM and pIBM in OpenFSI, and can easily choose an appropriate IB scheme according to the specific problems.

2.5. Immersed boundary method-based spatial decomposition and data communication

2.5.1. Spatial decomposition

Both Palabos and LAMMPS adopt spatial decomposition for parallel computing. In OpenFSI, the spatial decomposition is still applied to both fluid and structure domains. To ensure the correct interpolation of velocity and force in IBM, the consistent spatial decomposition is required that each processor can access both fluid and solid nodes within the same sub-domain. In the current framework (cf. Fig. 4), it includes three steps: (1) Initialization of sub-domains in LAMMPS; (2) Mapping domain decomposition information to Palabos; and (3) initialization of sub-domains in Palabos. Next, we will introduce this process in detail.

- Step 1: Initialization of domain in LAMMPS
  Two styles of decomposition are provided in the LAMMPS: *brick* and *tiled*. For *brick* style, the domain decomposition is fulfilled by partitioning the simulation box into a regular 2d or 3d grid of bricks. One processor masters one brick, and each processor communicates with its Cartesian neighbors in the grid to extract needed information. In the *tiled* style, the simulation box is partitioned into non-overlapping rectangular-shaped "tiles" with different sizes. In current work, the *brick* style is adopted to avoid the complex pattern of neighboring processors by applying the command *comm_style brick* in the input file of *LAMMPS*. For sake of clarity, only 2d grid is illustrated here. Fig. 4 shows a simple example that a simulation box is partitioned into 9 bricks and 9 processors are assigned to manipulate the domain. Specifically, the green brick i with four vertexes (i_1, i_2, i_3, i_4) is assigned to the processor numbered P_n.
- Step 2: Mapping domain decomposition information to Palabos
  To simplify communication of information, the physical simulation boxes Γ are the same in both LAMMPS and Palabos with the same coordinate system. Therefore, the mapping of domain decomposition information Σ can be implemented by the coordinate mapping $\{\Sigma|(i_1, i_2, i_3, i_4) \to (j_1, j_2, j_3, j_4), i_k \in \Gamma, j_k \in \Gamma\}$.
- Step 3: Initialization of domain in Palabos
  In the Palabos, the so-called *multi-block* structure is employed. The simulation box is divided into regular blocks, one per processor. To complete the parallelization, it is necessary to contain a loop which runs over the indices of the multi-block that each processor can recognize the domain that it will be responsible for. For instance, from the coordinate mapping, Palabos obtains the information that the processor P_n is responsible for the block j characterized by (j_1, j_2, j_3, j_4). And over the loop, the block j is assigned to the processor P_n.
  In summary, the spatial decomposition can be simply described by the following pseudo algorithm, as presented in Fig. 5.

For the individual part, parallel communication is realized through ghost technology. In LAMMPS, each solid node (particle) has ghost nodes in its neighboring bricks. The ghost nodes are replications of solid nodes with attributions: coordinates, velocity, force, etc. Therefore, when these solid nodes need to interact with their counterparts in neighboring sub-domains, they can access the properties directly through ghost nodes. No interprocessor communication is requested. Similarly, the ghost layer is adopted in Palabos, as shown in Fig. 4 near the boundaries of each block with gray color of width Δ_e. In OpenFSI, the width of the ghost layer depends on the IBM interpolation stencil. For

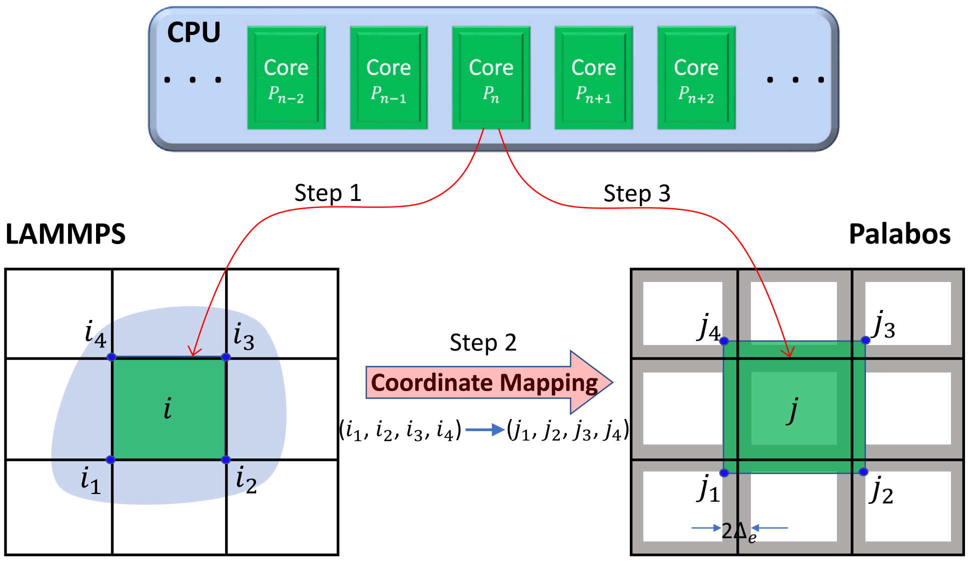


Fig. 4. (a) Spatial decomposition scheme for parallel computing. The numbers represent the rank of processors. The gray regions denote the ghost layers in Palabos. (b) Declaration of pointer *LammpsWrapper* to access all the members from LAMMPS, and corresponding wrapper functions to manipulate the members in LAMMPS. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Algorithm 1: Immersed boundary-based spatial decomposition

```
Run LAMMPS
Read simulation box $\Gamma$
Partition $\Gamma$ into $N$ regular bricks          #comm_brick
Run interface                                  #contained in file: latticeDecomposition.hh
Pass bricks information to Palabos
Run Palabos
Create $N$ blocks with the information from LAMMPS
For loop $m$ in $N$ blocks
        Assign block information to processor numbered $n$
End loop
Initialization parallelism in Palabos
```

Fig. 5. Pseudo algorithm for immersed boundary-based spatial decomposition of LAMMPS and Palabos.

example, for the 4-points stencil, the width of ghost layer Δ_e is larger than 2-points that enough fluid nodes can be accessed in each individual processor. With Δ_e, the coordinate mapping is realized by, for example, $(j_{1x} = i_{1x} - \Delta_e, j_{1y} = i_{1y} - \Delta_e)$ and $(j_{2x} = i_{2x} + \Delta_e, j_{2y} = i_{2y} - \Delta_e)$, where the subscripts x and y represent the coordinates in $x-$ and $y-$ directions, respectively. One thing should be noted that if the nodes distributed in the domain are extremely nonuniform, the efficiency will be significantly affected as the current domain decomposition is a regular grid-based method. For example, if the number of the nodes in one processor is small, while that in other processor is large, the processor with a small number of nodes is always waiting for the processor which has a large number of nodes.

2.5.2. Data communication

Because the structure is updated within the LAMMPS solver, the data information such as the coordinate, velocity, and force are analyzed and stored in LAMMPS. While the fluid flow properties including velocity and body force are obtained in Palabos. To exchange the data, we provide an interface named IB interface to collect all the information from both LAMMPS and Palabos, leaving no intrusion for the individual solver. The data transfer flow can be described in detail as shown in Algorithm 2, Fig. 6. There are two parts in the IB interface, one having to do with the interpolation of velocity, and the other is spreading of the FSI force, which corresponds to the IBM scheme (cf. Fig. 3(b)).

3. Validation of the numerical method

3.1. Validation of the LM

Although the validation of LM has been shown by Zhang [69], the implementation of LM in LAMMPS remains to be validated. The validations of LM are fulfilled by deflections of a 2D and a 3D beam with rectangular cross-section under uniformly distributed tractions. The results are compared with the corresponding FEA results in ABAQUS [101]. In the following examples, the unit system is: mm for length, mN for force, and kPa for stress. And in the simulations, the shear modulus is set as $\mu^s = 1$ kPa, the Lame constant is $\lambda = 100\mu$, which corresponds to Poisson's ratio 0.495.

First, a 2D beam with length 80 and width 20 is deflected under traction $t = [0, 0.025]^T$. The left side of the beam is fully fixed and the traction is applied on the right side (cf. Fig. 7(a)). The ABAQUS simulations are using a 4-node bilinear element with reduced integration, hourglass control, and hybrid with constant pressure (CPE4RH). We demonstrate the simulations with

```
Algorithm 2: Data communication between LAMMPS and Palabos

Run LAMMPS
Obtain coordinates of structure nodes $x(X,t)$, and pass them to IB interface.
Run Palabos
Obtain the velocity of fluid nodes $u(x,t)$, and pass them to IB interface.
Run IB interface                                  #contained in files: ibm2D.hh or ibm3D.hh
Interpolate the velocity $u^s$ at location $x(X,t)$, and return them to LAMMPS.
Run LAMMPS
Calculate the FSI force $F$, and pass them to IB interface
Run IB interface                                  #contained in files: ibm2D.hh or ibm3D.hh
Spread FSI force $F$ to the fluid nodes $f'$ at location $(x,t)$, and pass them to Palabos.
Run Palabos
Use the body force nodes $f'$ to advance fluid field.
Run LAMMPS
Use FSI force $F$ to update the new location of structure.
```

Fig. 6. Pseudo algorithm for data communication between LAMMPS and Palabos solvers.

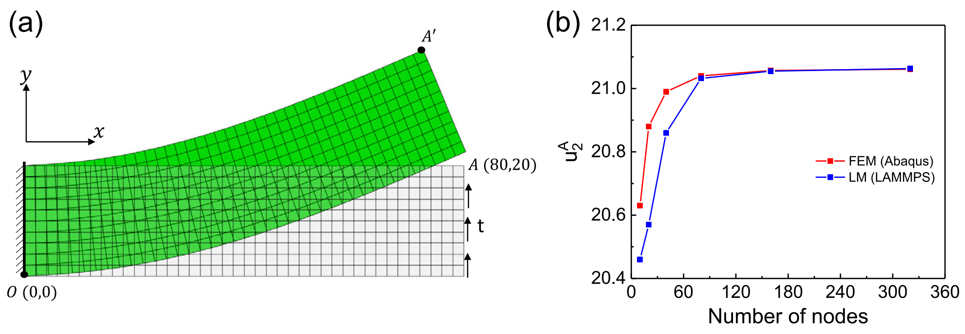


Fig. 7. (a) Initial and deformed configurations of the 2D beam. (b) Comparison of displacement point A in y-direction between LM and FEA in ABAQUS.

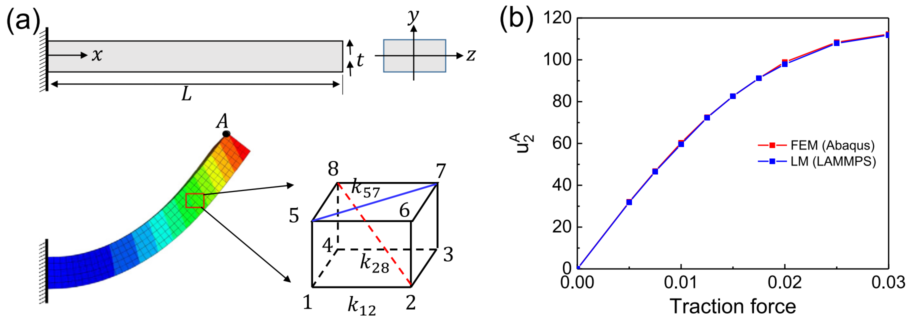


Fig. 8. (a) Schematic of 3D beam with rectangular cross-section. The traction load is applied in y-direction. The deformed state of beam is colored by the displacement in y-direction. The zoom-out displays the 3D lattice structure. (b) Comparison of displacement of point A in y-direction between LM and FEA under different loads. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

different element resolutions and monitor the displacement of point A in y-direction u_2^A. As shown in Fig. 7(b), the horizontal axis denotes the number of nodes used to discretize in length (x-direction). The LM implemented in LAMMPS adopts the same mesh as that in ABAQUS, and runs with the potentials we introduced in the above Section 2. We find that when the element resolution is low, an obvious discrepancy is obtained between LM and FEA due to the shear locking. However, with the increase of the resolution, LM can reproduce the same result with that in FEA.

We next adopt the LM to conduct the 3D beam deflection shown in Fig. 8(a). The left side in y-z plane is fixed, and the right side is applied with uniformly distributed traction $t = [0, t_0, 0]^T$. The lengths in x, y, z directions are 160, 20, and 40, respectively. The nodes in the x-direction is 240, which is confirmed to be mesh-converged in the 2D case. Also, we have the same setup in FEA model, and the ABAQUS simulations employ 8-node linear brick element with reduced integration, hourglass control, and hybrid with constant pressure (C3D8RH). We vary the traction force magnitude t_0 and calculate the deflection of point A in the y-direction, which locates at the center of the upper edge of the right side (c.f. Fig. 8(a)). It is found that the comparison of the results between LM and FEA exhibit excellent agreement. The 2D and 3D benchmarks demonstrate that our LM can reproduce the same results as those in FEA. The membrane model used in current simulations has been validated by our previous works [9,40].

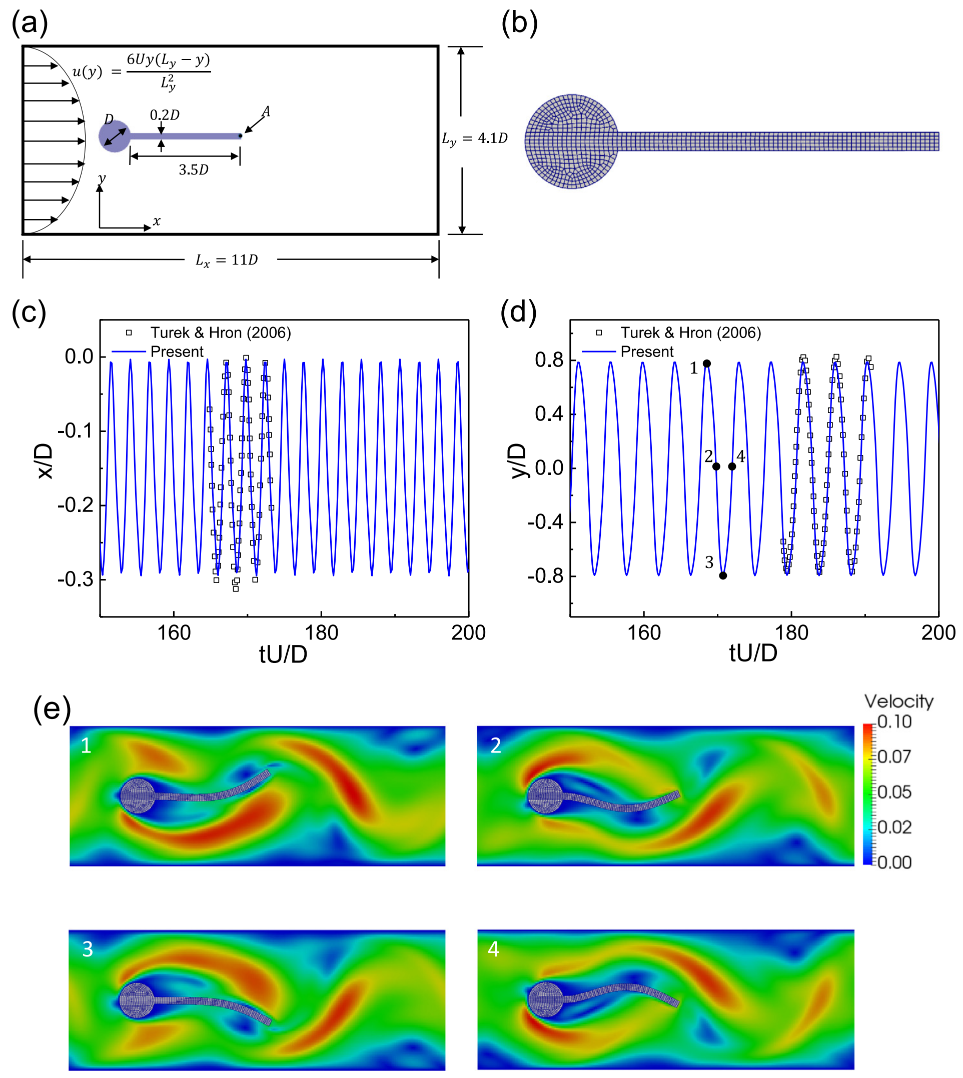


Fig. 9. (a) Schematic of the model for flow induced flapping of 2D beam behind a cylinder. (b) Discretization of the immersed structure (combination of cylinder and beam). Variation of (c) x- and (d) y-direction positions of the point A in the center of free end of beam. Open symbols represent the results from Turek et al. [102]. (e) Velocity field in four different time points marked in (d).

3.2. Validation of FSI: flow-induced flapping of an elastic 2D beam behind a cylinder

To validate our 2D FSI model, we adopt the classical benchmark problem: flow-induced flapping of a beam behind the cylinder, which was proposed by Turek et al. [102]. The schematic is shown in Fig. 9(a). The lengths in this problem are normalized by the diameter of the cylinder $D = 20$ m. The beam with length $3.5D$ and width $0.2D$ behind the cylinder is placed in a rectangular domain with length $11D$ and width $4.1D$. The center of the cylinder is fixed at $(2D, 2.05D)$. The top and bottom sides of the domain are applied non-slip walls and a parabolic velocity profile $u(y) = \frac{6Uy(L_y-y)}{L_y^2}$ is applied in the inlet, where $U = 0.005$ m/s. A constant pressure boundary condition is set at the outlet. In this case, the cylinder is fixed and the beam is allowed to move freely. They are a combination that the beam is clamp-mounted rather than simply attached to the cylinder (cf. Fig. 9(b)). The beam is discretized into 762 quadrilateral elements and 866 nodes. The Reynolds number is set as $Re = UD/\nu = 100$, and density ratio $\rho^s/\rho^f = 10$. The kinematic viscosity of the fluid $\nu = 10^{-3}$ m^2/s and the fluid density is $\rho^f = 10^3$ kg/m^3. The nondimensional elasticity is defined to be $E^* = E/(\rho^f U^2) = 1.4 \times 10^3$, and the Poisson's ratio $\nu_s = 0.4$. The discretization of fluid mesh and averaged solid mesh sizes are $\Delta x = 0.05D$ and $\Delta s \approx \Delta x$, respectively. The time step is set $\Delta t = \Delta x$.

The time-varying positions of point A in the center of the free end are shown in Fig. 9(c) and (d) in x- and y-direction, respectively. We also present the results from Turek et al. [102] for comparison. From the figure, we can see that, after reaching a steady flapping state, both the flapping period and the amplitude of beam in the current simulation have a good agreement with those from Turek et al. [102]. This demonstrates that our method can reproduce the same motion of beam as previous work. Finally, we present the velocity field in a flapping period. Fig. 9(e) gives four velocity contours with the time points marked as 1 to 4 in Fig. 9(d). The position of beam in these four typical time points is also given. A movie showing the dynamic motion of the beam in the fluid is provided in *Supplementary Material*. To further confirm the accuracy, we calculate the Strouhal number, which is defined as $St = fU/D$ (f is the flapping frequency), and averaged drag coefficient $C_D = \bar{f}_x/(0.5\rho^f U^2 D)$. They are listed in Table 1. In addition to the work of Turek et al. [102], we present more previous studies involving the same FSI problem. And from Table 1, we can find our method has quantitative consistence with those previous works.

Table 1
Comparison of amplitude of point A, Strouhal number and drag coefficient with previous works.

Studies	Amplitude	Strouhal number (St)	Drag coefficient C_D
Turek et al. [102]	0.83	0.19	4.13
Tian et al. [35]	0.78	0.19	4.11
Lin et al. [103]	0.81	0.19	4.10
Present	0.79	0.19	4.10

3.3. Validation of FSI: deformation of an elastic capsule in bi-shear flow

To validate the accuracy of modeling a membrane immersed in fluid flow using OpenFSI, we adopt a capsule as a membrane model and it is placed in a bi-shear flow. We calibrate the deformation of the capsule. First, the spherical capsule is discretized using a MATLAB code implemented by Persson [104]. The Lagrangian mesh of the capsule is approximately uniform and the size is about $\Delta s \approx 250$ nm. The total nodes of the capsule is 726. We place the capsule in the center of a channel with dimension $L_x \times L_y \times L_z = 10D_p \times 10D_p \times 10D_p$. $D_p = 2$ μm is the diameter of the capsule. We apply periodic boundary conditions in the x and z directions. y-direction is bounded by two flat plates. The bi-shear flow is driven by the moving of flat plates with opposite velocities of magnitude $U_0 = 1$ μm/s (cf. Fig. 10(b)). The time step is set $\Delta t = 2 \times 10^{-4} L_y/U_0$, and the time is nondimensionalized $t^* = \frac{2U_0}{L_y} t$. Under the bi-shear flow, the capsule will deform and make tank-treading motion [105]. We use two parameters to quantify the deformation and the motion of the capsule. One is the Taylor parameter D_{xy}, which is defined as $D_{xy} = (a-b)/(a+b)$, where a and b are semi-major and semi-minor axes of the deformed capsule, respectively. The other one is the inclination angle of the capsule between the major axis of the capsule and flow direction (x-direction) as shown in Fig. 10(b). The deformability is characterized by the Capillary number $Ca = \mu U_0 D_p/\mu_s L_y$, where $\mu_s = 1$ μN/m is the 2-dimensional shear modulus of the capsule. We conduct two cases with different deformabilities $Ca = 0.0125$ and $Ca = 0.125$ through tuning the fluid density μ. The results of Taylor parameter and inclination angle are compared with those in [105]. We find that our simulation results are in good agreement with the previous computational study in [105]. It further confirms that the modeling of the membrane is accurate enough using our proposed method.

3.4. Validation of FSI: uniform flow passing a fixed cylinder

A case of uniform flow passing a fixed cylinder in a 3D rectangular channel is further studied. As shown in Fig. 11(a), a cylinder with diameter $D = 0.5$ cm, length $L = 2$ cm in z-direction is placed inside a rectangular channel with size $L_x \times L_y \times L_z = 2 \times 8 \times 2$ cm^3. The center of the cylinder in x–y plane locates at $(L_x/2, L_y/8)$. The cylinder is discretized into hexahedron meshes with total of 30134 nodes and 27660 elements (cf. Fig. 11(a)). A constant uniform flow with velocity U_0 is applied at the inlet $y = 0$. The outlet ($y = 8$) boundary condition is stress-free. No-penetration boundary condition is adopted at the top ($x = 2$) and bottom ($x = 0$) boundaries of the channel. The front ($z = 0$) and back ($z = 2$) boundaries are non-slip and non-penetration walls. The Reynolds number is defined as $Re = \rho^f U_0 D/\mu$, where $\rho^f = 10^3$ kg/m^3 and $\mu = 1$ kg/(m · s). Two cases $Re = 50$ and $Re = 100$ are considered here, which corresponds to no vortex shedding and vortex shedding flow regimes, respectively. Fig. 11(b) gives the snapshots of velocity fields in these two cases. We can find when $Re = 50$, the flow is steady-state and there is no vortex forming and shedding from the trail of the cylinder. While in the case $Re = 100$, the flow becomes unstable and the vortexes form and shed from the trail. To quantify the shedding frequency, we examine the lift and drag coefficients. They are defined as $C_L = F_x/(\frac{1}{2}\rho^f U_0^2 DL)$ and $C_D = F_y/(\frac{1}{2}\rho^f U_0^2 DL)$, respectively. F_x and F_y are the forces exerted on the cylinder, and these forces can be calculated directly in LAMMPS. We show the drag coefficients C_D for these two cases $Re = 50$ and $Re = 100$ in Fig. 11(c). We can see that for case $Re = 50$, the drag coefficient reaches a plateau $\bar{C}_D = 1.27$ after a short initial transition. It means that the flow quickly reaches a steady-state. However, in the case $Re = 100$, the drag coefficient quickly reaches a steady oscillatory state with an averaged value about $\bar{C}_D = 1.18$. The oscillatory is caused by the alternating vortex shedding and the vortex shedding frequency corresponds to the oscillation frequency in lift force. The frequencies are estimated using FFT power spectrum to be $f^{C_D} = 80.2$ Hz and $f^{C_L} = 39.4$ Hz for drag and lift coefficients, respectively. Based on shedding frequency calculation, we can obtain the Strouhal number of the vortex shedding case $St = 0.196$. All the values, including averaged drag coefficient and Strouhal number are quite close to the expected values of corresponding results in [106] with the case of flow passing a stationary, smooth, and infinitely long cylinder.

4. Results: Application of the OpenFSI for different FSI problems

4.1. Free-falling of single and multiple rigid spheres

The first application is the free-falling of a single 3D sphere in quiescent fluid. The fluid domain is $L_x \times L_y \times L_z = 5 \times 15 \times 5$ cm^3, and the fluid density is $\rho^f = 10^{-3}$ g/cm^3. The density and diameter of the sphere are $\rho^s = 1$ g/cm^3 and $D = 0.5$ cm, respectively. And the Young's modulus of the sphere is $E = 10000$ dyn/cm^2. The boundaries in x- and z-directions are non-slip walls, and a periodic boundary condition is applied in the y-direction. The sphere is initially placed at $(1, 1, 1)$ and its movement is driven by gravity of $g = 9.80$ m/s^2. The fluid mesh size is $\Delta x = 0.025$ cm and the sphere is discretized into 2078 hexahedron elements and 2870 nodes with nearly uniform mesh size $\Delta s \approx \Delta x$.

The snapshots shown in Fig. 12(a) describe the free-falling process of the sphere. There are two vortexes formed on two sides of the sphere at the initial state, and these vortexes will be stretched with the falling of sphere. To quantify the motion of the sphere, the displacement and velocity are calculated and compared with the theoretical results, which are given in Fig. 12(b). An excellent agreement is observed between present results and theoretical predictions. Thanks to the solution of this problem without much complex phenomena, the purpose of this example is to merely demonstrate the capability of the OpenFSI that can handle large displacement of nearly rigid structures. Additionally, to mimic the deposition of granular particles in the industry, we show that the current method can also simulate the falling of multiple spheres. It is presented in Fig. 13. 16 identical spheres are placed in the center of the channel with $L_x \times L_y \times L_z = 10 \times 10 \times 5$ cm^3 in a crossed pattern. At the initial state, the spheres can fall freely under gravity. However, then they will interact with each other, leading to irregular falling cascade.

4.2. Flow passing 3D flexible plate

In this part, we demonstrate the simulation results about a 3D flexible plate that deforms under a cross-flow. This case is usually chosen as a model to study the deformation of aquatic plants by flow of water. The model is shown in Fig. 14(a). A flexible plate with length L, width b and thickness h is vertically (length in

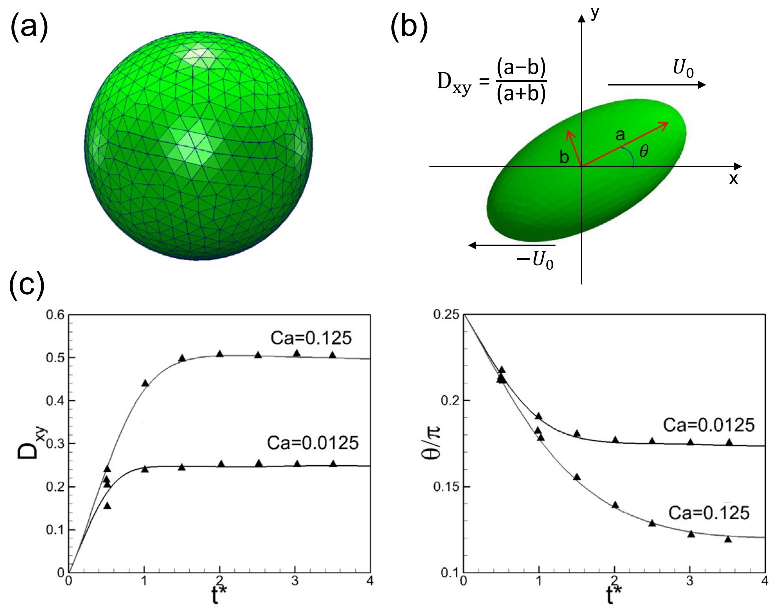


Fig. 10. (a) Sketch of mesh used to represent the capsule. (b) Schematic of deformation of a single capsule in bi-shear flow. (c) The evolution of Taylor parameter and inclination angle.

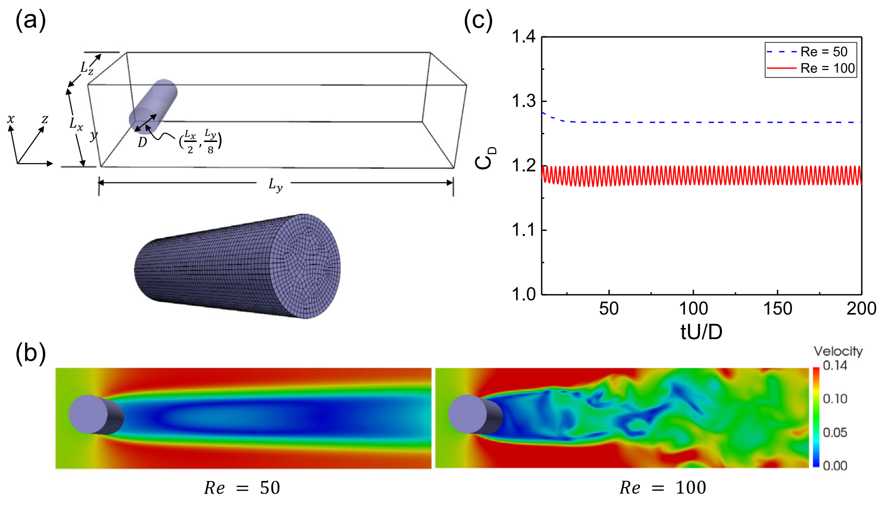


Fig. 11. (a) Schematic of uniform flow passing a fixed cylinder in rectangular channel and mesh of the cylinder. (b) Velocity fields for the slice of the middle-plane in z-direction for cases $Re = 50$ and $Re = 100$. (c) Evolution of drag coefficient for cases $Re = 50$ and $Re = 100$.

z-direction) placed in a rectangular channel of size L_x, L_y and L_z. One end of the plate is clamp-mounted at $(L_x/2, L_y/5, 0)$ and the other end is free to move. We use the width to normalize the length: $L = 5b, h = 0.2b, L_x = 4b, L_y = 21b$ and $L_z = 8b$, where $b = 2$ m. The inlet ($y = 0$) is applied with a parabolic velocity profile and average value U_0 in y-z plane; the outlet ($y = L_y$) is set with constant pressure boundary condition. The flow in the x-direction is periodic. The Reynolds number here is defined as $Re = \rho^f U_0 b/\mu$, where $\rho^f = 10^3$ kg/m^3 and $\mu = 1$ kg/(m $\cdot$ s). There are two cases considered in present work: (1) $Re = 60$; and (2) $Re = 100$ that correspond to $U_0 = 0.03$ m/s and 0.05 m/s, respectively. For the flexible plate, we have dimensionless elasticity $E^* = E/\rho^f U_0^2 = 14820.3$, Poisson's ratio $\nu_s = 0.4$ and density ratio $\rho^s/\rho^f = 1.5$. In the simulation, the fluid mesh $\Delta x = 0.25b$. The plate is discretized into 2244 nodes and 1500 hexahedron elements.

First, the velocity and vorticity fields for these two cases are shown in the Fig. 14 (b) and (c), respectively. From these figures, we can find when $Re = 60$, the velocity field is regular and there is small oscillation happening near the free end of the plate. Weak vortex can be observed, shedding from the free end of the plate. When it comes to large $Re = 100$, the velocity field becomes chaotic, especially for the regions near the plate. Many vortexes form in the flow field (cf. *Supplementary Material*). Then we quantitatively study the deformation of the plate and drag coefficients. The drag coefficient has the definition $C_D = F_y/(0.5\rho^f U_0^2 bL)$, where F_y is the hydrodynamic force exerted on the plate in the flow direction (y-direction). As shown in Fig. 14(d), the drag coefficients oscillate with an averaged value, but the oscillation is irregular, which is not the same as that in the flow passing a fixed cylinder where the oscillation is sinusoidal. Nevertheless, we can find the averaged value of C_D in the case of $Re = 100$ is smaller than that in the case of $Re = 60$. Then we list these averaged values in the Table 2, along with the deflection of the free end of plate in y- (D_y) and z-direction (D_z). The drag coefficients give comparative results with the previous work by Tian et al. [35].

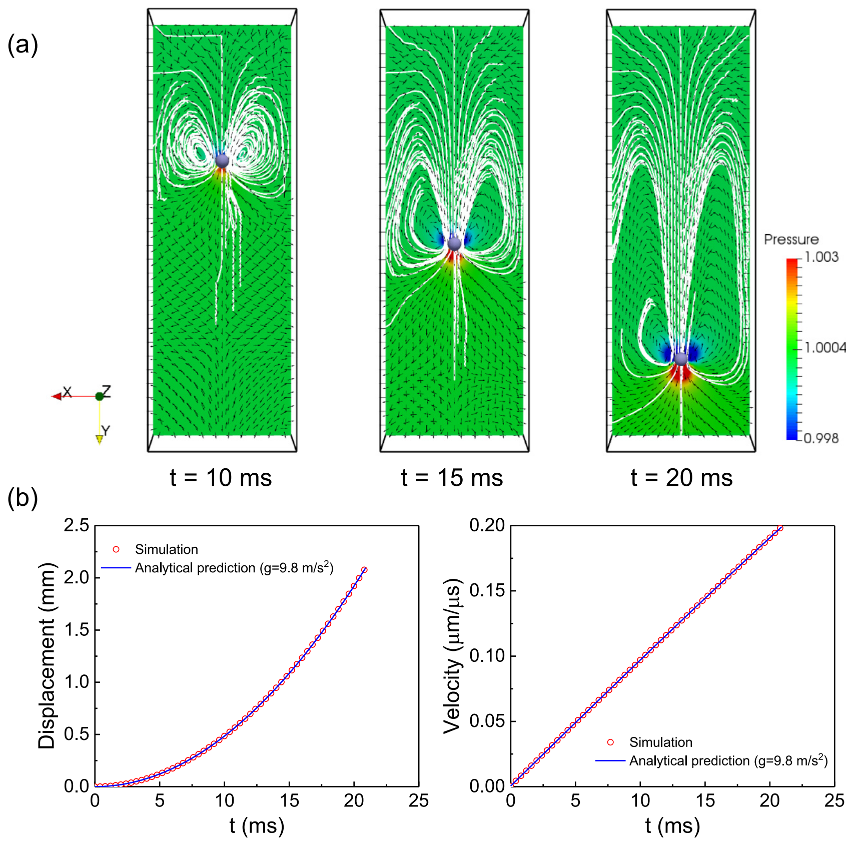


Fig. 12. (a) Snapshots for free-falling process of a single sphere at time $t = 10, 15, 20$ ms. (b) Comparison of displacement and velocity of the falling sphere between our simulation results and theoretical results.

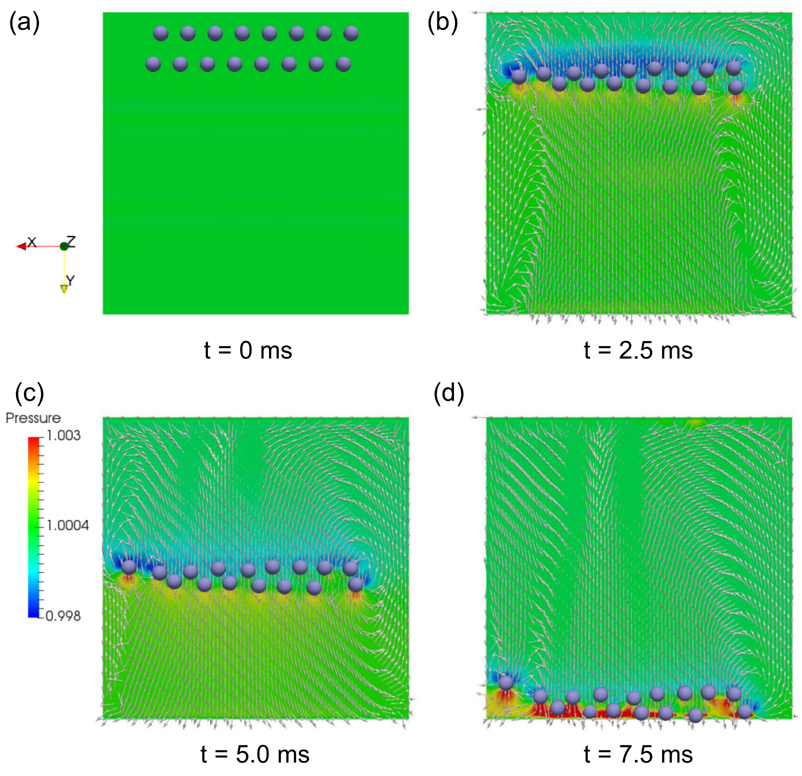


Fig. 13. Snapshots for free-falling process of multiple rigid spheres at time $t = 0, 2.5, 5.0$ and 7.5 ms.

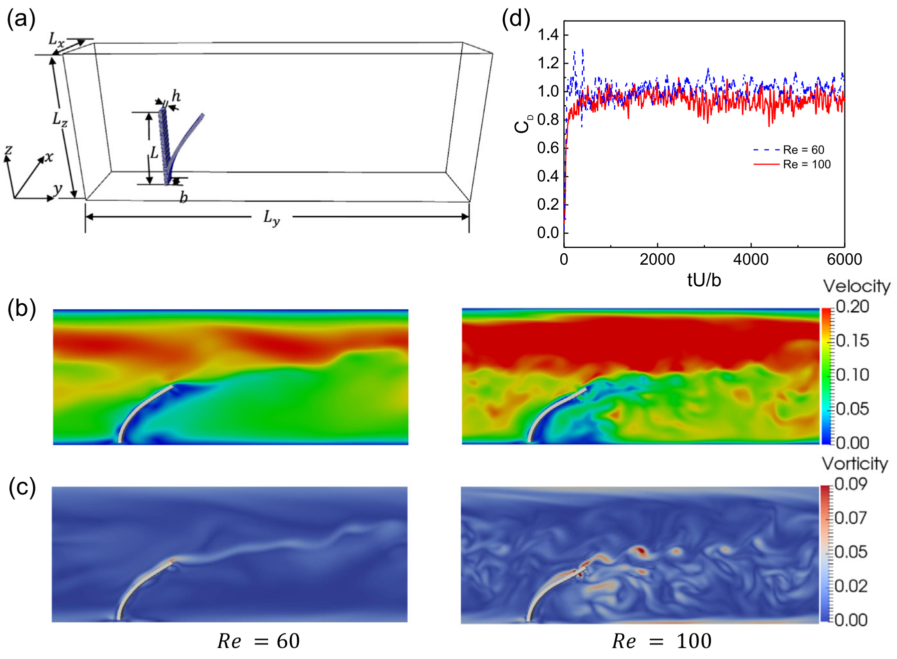


Fig. 14. (a) Schematic of the computational model for bending of plate in the cross-flow. The initial and deformed configurations of plate are shown. Snapshots to show the (b) velocity and (c) vorticity fields. (d) Evolution of drag coefficient of the plate in cross-flow for different Reynolds number $Re = 60$ and $Re = 100$.

Table 2
Bending of plate in cross-flow: deflection of plate in y- and z-directions, and drag coefficients.

Case	Deflection D_y/b	Deflection D_z/b	Drag coefficient C_D
$Re = 60$	3.12	1.58	1.04
$Re = 100$	3.53	2.06	0.93

And readers interested in the flow passing plate with higher Reynolds number are referred to [35].

4.3. Passing of rigid spheres over dam in uniform flow

To demonstrate that we can model different types of solids simultaneously in the flow, we conduct the study of the rigid spheres over a fixed dam in the uniform flow with velocity U_0. As shown in Fig. 15(a), a rectangular dam with length (x-direction) $L = 40$ cm, width (y-direction) $W = 8$ cm and height (z-direction) h is placed in the fluid domain of $L_x \times Ł_y \times L_z = 48 \times 240 \times 48$ cm^3. The center of the front bottom edge for the dam locates at $(4, L_y/2, 0)$. The dam is discretized into hexahedron elements and there are total 140 rigid spheres with diameter $D_s = 1$ cm initial uniformly set in x–z place at $y = 100$ cm. The Reynolds number here is defined as $Re = \rho^f U_0 L/\mu = 20$, where $\rho^f = 10^3$ kg/m^3, $\mu = 1$ kg/(m · s) and $U_0 = 5$ cm/s. There are two cases considered here $h = 1/3L_z$ and $h = 2/3L_z$. The instantaneous sphere distributions for dams with different heights at the same time are shown in Fig. 15(b) and (c), respectively. We find when the dam is short ($h = 1/3L_z$), the distribution of spheres in the flow is nearly uniform. While the spheres accumulate in front of the tall dam with a height $h = 2/3L_z$.

4.4. Active motion of a micro-swimmer with filamentous tails

As the last numerical case, we demonstrate the possibility of modeling the swimming of flagellated microorganisms at low Reynolds number, which inspires the design and production of artificial micro-swimmers [107]. A simple model is shown in Fig. 16(a) and (b), where the micro-swimmer is considered as a rigid spherical head attached with one ($N = 1$) or multiple ($N = 2$) filamentous tails (1D LM). The active motion is driven by the oscillation of the rigid head in the z-direction with motion $X = A\sin(\omega t)$, where X is the center of the bead, A is the oscillation amplitude and ω is the oscillation frequency. The filamentous tail is clamp-mounted at the trail of the bead and modeled as consecutive small beads connected with linear springs and angle springs. The fluid is water with $\rho^f = 10^3$ kg/m^3 and viscosity $\mu = 1$ kg/(m · s). For the microorganism, to have universal applications, we use the nondimensional values in lattice units, which can be easily converted to specific units according to the problem. The diameter of the spherical head is set $D = 100$, the length of filament is $L = 4D = 400$, and the discretized spring length is 20. For the multiple tails, the distance between them is $d = 20$. The oscillation amplitude is $A = 50$ and the period is $T = 2000$. The strength of spring $k_s = 1000$ and bending strength is set $k_b = 5000$.

The drag coefficients $C_D = F_x/(0.5\rho^f AD^2/T)$ are presented in Fig. 16(c). We find for the case $N = 1$, the drag coefficient oscillates with averaged value nearly 0, and the oscillation amplitude is very small. However, the oscillation of the drag coefficient is irregular and has a negative averaged value for the case $N = 2$. This means the micro-swimmer with one tail has extremely small propulsion with the active oscillation of the spherical head, while there is a thrust generated in the case with multiple tails. It is consistent with the previous study [108] that multiple tails will enhance the performance of micro-swimmer. The velocity field for the case $N = 2$ is also presented in Fig. 16(d). A movie presents the dynamic motion of micro-swimmer with two tails and the flow field is given in *Supplementary Material*.

5. Computational efficiency of OpenFSI

To test the computational efficiency of OpenFSI, we choose a large system with 2000 red blood cells in the blood flow (cf. Fig. 17(a)). To exclude the dependency of the distribution of particles in the space, the RBCs are randomly packed inside the rectangular channel. The computational domain is 72 × 144 ×

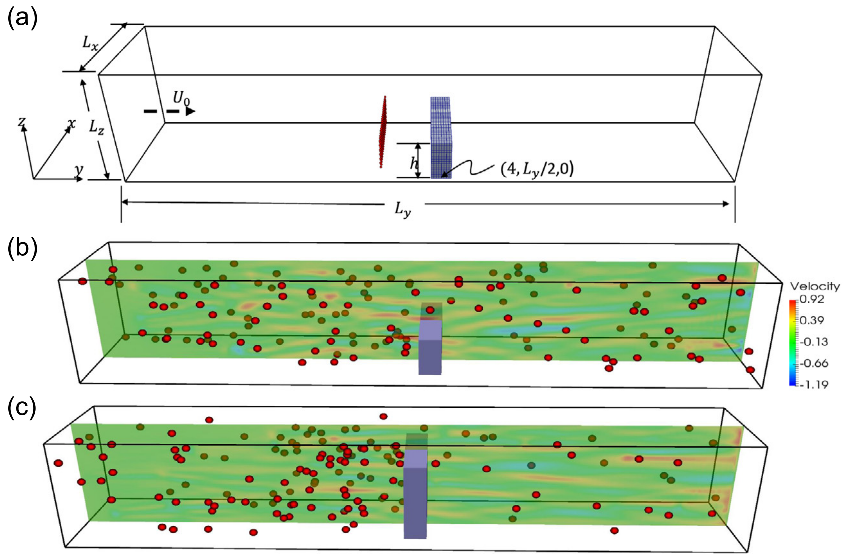


Fig. 15. (a) Schematic of rigid spheres passing a fixed dam in uniform flow. Distribution of spheres in the flow for dams with different heights: (b) $h = 1/3L_z$; and (c) $h = 2/3L_z$. The contour shows the velocity field in the slice of middle-plane in x-direction.

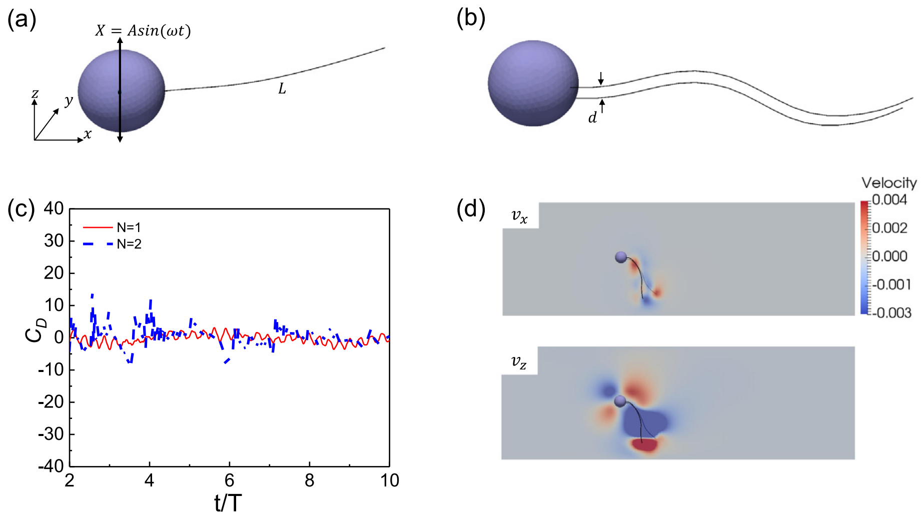


Fig. 16. Models of micro-swimmer with (a) one and (b) two tails that oscillate in the quiescent fluid flow. (c) The drag coefficients of micro-swimmer in x-direction for cases $N = 1$ and $N = 2$. (d) The velocity field of v_x and v_z around the micro-swimmer for case $N = 2$.

72 μm^3, and the fluid mesh is $\Delta x = 250$ nm, which corresponds to the total fluid mesh grids $288 \times 576 \times 288 = 47,775,744$. The fluid is water with $\rho^f = 10^3$ kg/m^3 and viscosity $\mu = 1$ kg/(m·s). The diameter of red blood cells is 7.82 μm and it is numerically discretized into the membrane model using the same method for the capsule. 2000 red blood cells are discretized into 6, 572, 000 nodes, 19, 704, 000 bonds, 13, 136, 000 angles and 19, 704, 000 dihedrals. The flow is a simple shear flow that is driven by the moving of upper plate in the z-direction with velocity $U_0 = 0.1$ μm/μs. We test two LBM solvers: (1) Palabos; and (2) *fix lb* package in LAMMPS. *fix lb* package is a fluid dynamics solver based on LBM, which is embedded within LAMMPS [109] by Mackay et al. [70]. The case is running in two different computer architectures: (1) *Intel Xeon Knights Landing (KNL) (CPU model: Intel Xeon Phi 7250)* and (2) *Skylake (CPU model: Intel Xeon Platinum 8160)*. We run each case for 4000 time steps, corresponding to 0.0164 s, and collect the simulation times together. To separately examine the scalabilities of each component of OpenFSI, we consider the simulation times of fluid part, structure part, and IB interface part in Fig. 17(b), (c), and (d), respectively.

Comparing the simulation times of these three parts running in the same computer architecture, we find that the time for fluid part occupies the most portion of the total computational cost. For example, for Palabos package running in *Skylake*, the time of the fluid part is up to 68%, while the times used by the structure and IB interface are as low as 13% and 17%, respectively. Others should be taken by the input and output of data files. The distribution of the consumption of simulation time demonstrates that the fluid part is dominant in the whole simulation. Specifically, we take two fluid solvers, e.g., *fix lb* and Palabos, into consideration to compare the efficiency as shown in Fig. 17(b). It is found that the Palabos is nearly 1.5 times faster than the *fix lb* package for the same computer architecture, which reflects that the LBM solver in Palabos is more efficient than that implemented in LAMMPS. This may be caused by the different methods adopted to solve the streaming process as we discussed in Section 2. Furthermore, for different computer architectures, the *Skylake* system outperforms the *KNL* system in terms of the computation speed due to the higher CPU clock frequency. Also, the scalability in *Skylake* system is better than that in *KNL* system, which is

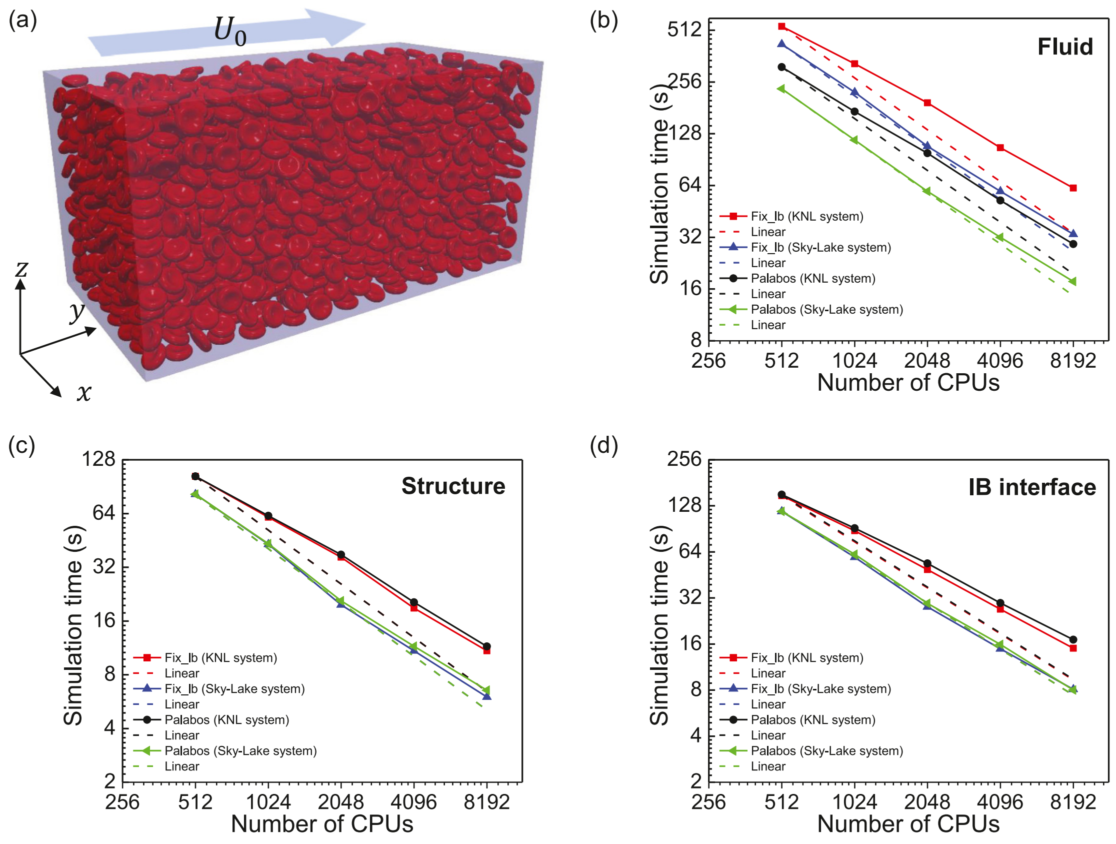


Fig. 17. (a) Model of large scale blood flow simulations with 2000 red blood cells. Separate simulation times of (b) fluid part, (c) structure part, and (d) IB interface part for running 2000 time steps in different systems with different solvers.

attributed to the worse communication between nodes in *KNL* system. Therefore, it is highly recommended to use the Palabos as the fluid solver in *Skylake* system for better efficiency. The *fix lb* package can be an alternative option regarding the simplicity, since it is embedded within the LAMMPS framework already. From Fig. 17(c), we find that there is no significant difference of the simulation times comparing different fluid solvers, while the *Skylake* system still outperforms the *KNL* system in terms of solving the structure part in both simulation time and scalability, similar to that in fluid part. The simulation time of IB interface demonstrated in Fig. 17(d) also displays irrelevance to the fluid solver. While the scalability should be noted that it is nearly linear up to 8192 cores in the *Skylake* system, despite the poor performance in *KNL* system. It confirms that our IB interface is very efficient in terms of the implementation in both *fix lb* and Palabos packages. All these results demonstrate that OpenFSI is a highly efficient and well-parallelized simulation package for solving large-scale FSI problems.

6. Conclusion

Present study offers a continuum-informed and particle-based FSI simulation package, OpenFSI. Within OpenFSI, the LBM and LM are used to solve fluid dynamics and structure motions, respectively. And they are coupled together through IBM to satisfy the non-slip boundary condition on fluid–structure interface. To validate the accuracy of OpenFSI, several different cases are considered. First, the LM is confirmed to have the same accuracy with FEA for the benchmark problems of 2D and 3D deflections of a cantilever beam under traction. Then, two classical FSI problems are considered: (1) flow-induced flapping of 2D flexible beam behind the fixed cylinder; and (2) flow passing a 3D fixed cylinder. The simulation results given by OpenFSI are found in excellent agreement with previous studies. After these validations, the OpenFSI is adopted to study free-falling of rigid spheres and flapping of flexible plate in cross-flow. Furthermore, it also demonstrates the possibility to model complex systems such as rigid spheres over a dam under uniform flow, and swimming of micro-swimmers. Lastly, we consider a large system with thousands of red blood cells in blood flow under two different HPC systems to test the scalability of OpenFSI. We find that our method owns high efficiency, and the speedup is almost linearly increasing with the logarithmic value of the total number of CPUs up to 8192 cores for simulating large FSI problems. Therefore, the OpenFSI provides an alternative option to study large-scale and complex FSI problems, hence to facilitate the unveiling of underlying physical mechanisms. We have released the OpenFSI in Github (https://github.com/huilinye/OpenFSI), and more structure potentials and complex geometries are expected to be added in the near future.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

H.Y., Z.S., W.X. and Y.L. would like to thank the support by the National Science Foundation under the grant no. OAC-1755779. T.Z. acknowledged the support of the National Science Foundation under grant no. CMMI-1847149. H.Y., Z.S., W.X. and Y.L. are all grateful for the support from the Department of Mechanical

Engineering at the University of Connecticut. H.Y. and Z.S. were partially supported by a fellowship grant from GE's Industrial Solutions Business Unit under a GE–UConn partnership agreement. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Industrial Solutions or UConn. This research also benefited in part from the computational resources and staff contributions provided by the Booth Engineering Center for Advanced Technology (BECAT) at the University of Connecticut. Part of this work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by the National Science Foundation grant no. ACI–1053575. The authors also acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources (Frontera project and the National Science Foundation award 1818253) that have contributed to the research results reported within this paper.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cpc.2020.107463.

References

[1] D.E. Keyes, L.C. McInnes, C. Woodward, W. Gropp, E. Myra, M. Pernice, J. Bell, J. Brown, A. Clo, J. Connors, et al., Int. J. High Perform. Comput. Appl. 27 (1) (2013) 4–83.
[2] E. Wei, in: 2017 IEEE International Conference on Computational Electromagnetics, ICCEM, IEEE, pp. 338–340, 2017.
[3] B. Uekermann, H.-J. Bungartz, B. Gatzhammer, M. Mehl, A parallel, black-box coupling algorithm for fluid-structure interaction, in: Proceedings of 5th International Conference on Computational Methods for Coupled Problems in Science and Engineering, 2013, pp. 1–12.
[4] Ö. Babur, V. Smilauer, T. Verhoeff, M. van den Brand, Procedia Comput. Sci. 51 (2015) 1088–1097.
[5] T.-R. Teschner, L. Könözsy, K.W. Jenkins, Microfluid. Nanofluid. 20 (4) (2016) 68.
[6] L. Fan, J. Yao, C. Yang, D. Xu, D. Tang, Comput. Model. Eng. Sci. 114 (2) (2018) 221–237.
[7] J. Yan, W. Yan, S. Lin, G.J. Wagner, Comput. Methods Appl. Mech. Engrg. 336 (2018) 444–470.
[8] W. Yan, W. Ge, Y. Qian, S. Lin, B. Zhou, W.K. Liu, F. Lin, G.J. Wagner, Acta Mater. 134 (2017) 324–333.
[9] H. Ye, Z. Shen, Y. Li, Comput. Mech. 62 (3) (2018) 457–476.
[10] H. Ye, Z. Shen, Y. Li, IEEE Trans. Nanotechnol. 17 (3) (2018) 407–411.
[11] P. Balogh, P. Bagchi, Biophys. J. 113 (12) (2017) 2815–2826.
[12] H.-B. Deng, Y.-Q. Xu, D.-D. Chen, H. Dai, J. Wu, F.-B. Tian, Comput. Mech. 52 (6) (2013) 1221–1242.
[13] B.S. Connell, D.K. Yue, J. Fluid Mech. 581 (2007) 33–67.
[14] M.J. Shelley, J. Zhang, Annu. Rev. Fluid Mech. 43 (2011) 449–465.
[15] F. Boano, J.W. Harvey, A. Marion, A.I. Packman, R. Revelli, L. Ridolfi, A. Wörman, Rev. Geophys. 52 (4) (2014) 603–679.
[16] M.T. Odman, A.G. Russell, J. Geophys. Res. Atmos. 96 (D4) (1991) 7363–7370.
[17] N. Geneva, C. Peng, X. Li, L.-P. Wang, Parallel Comput. 67 (2017) 20–37.
[18] L.-P. Wang, C. Peng, Z. Guo, Z. Yu, Comput. Fluids 124 (2016) 226–236.
[19] C.W. Hirt, A.A. Amsden, J. Cook, J. Comput. Phys. 14 (3) (1974) 227–253.
[20] T.J. Hughes, W.K. Liu, T.K. Zimmermann, Comput. Methods Appl. Mech. Engrg. 29 (3) (1981) 329–349.
[21] W.K. Liu, D.C. Ma, Comput. Methods Appl. Mech. Engrg. 31 (2) (1982) 129–148.
[22] J. Fan, H. Liao, R. Ke, E. Kucukal, U.A. Gurkan, X. Shen, J. Lu, B. Li, Comput. Methods Appl. Mech. Engrg. 337 (2018) 198–219.
[23] T.E. Tezduyar, M. Behr, S. Mittal, J. Liou, Comput. Methods Appl. Mech. Engrg. 94 (3) (1992) 353–371.
[24] T.E. Tezduyar, S. Sathe, R. Keedy, K. Stein, Comput. Methods Appl. Mech. Engrg. 195 (17–18) (2006) 2002–2027.
[25] K. Takizawa, T.E. Tezduyar, Comput. Mech. 48 (3) (2011) 247–267.
[26] Y. Bazilevs, V.M. Calo, Y. Zhang, T.J. Hughes, Comput. Mech. 38 (4–5) (2006) 310–322.
[27] Y. Bazilevs, V.M. Calo, T.J. Hughes, Y. Zhang, Comput. Mech. 43 (1) (2008) 3–37.
[28] C.S. Peskin, J. Comput. Phys. 25 (3) (1977) 220–252.
[29] R. Mittal, H. Dong, M. Bozkurttas, F. Najjar, A. Vargas, A. Von Loebbecke, J. Comput. Phys. 227 (10) (2008) 4825–4852.
[30] L. Zhang, A. Gerstenberger, X. Wang, W.K. Liu, Comput. Methods Appl. Mech. Engrg. 193 (21–22) (2004) 2051–2067.
[31] L. Zhang, M. Gay, J. Fluids Struct. 23 (6) (2007) 839–857.
[32] W.-X. Huang, S.J. Shin, H.J. Sung, J. Comput. Phys. 226 (2) (2007) 2206–2228.
[33] W.-X. Huang, H.J. Sung, J. Fluid Mech. 653 (2010) 301–336.
[34] F.-B. Tian, H. Luo, L. Zhu, J.C. Liao, X.-Y. Lu, J. Comput. Phys. 230 (19) (2011) 7266–7283.
[35] F.-B. Tian, H. Dai, H. Luo, J.F. Doyle, B. Rousseau, J. Comput. Phys. 258 (2014) 451–469.
[36] H. Luo, R. Mittal, X. Zheng, S.A. Bielamowicz, R.J. Walsh, J.K. Hahn, J. Comput. Phys. 227 (22) (2008) 9303–9332.
[37] W.K. Liu, Y. Liu, D. Farrell, L. Zhang, X.S. Wang, Y. Fukui, N. Patankar, Y. Zhang, C. Bajaj, J. Lee, et al., Comput. Methods Appl. Mech. Engrg. 195 (13–16) (2006) 1722–1749.
[38] R.-N. Hua, L. Zhu, X.-Y. Lu, J. Fluid Mech. 759 (2014) 56–72.
[39] H. Ye, H. Wei, H. Huang, X.-y. Lu, Phys. Fluids 29 (2) (2017) 021902.
[40] H. Ye, Z. Shen, Y. Li, J. Fluid Mech. 861 (2019) 55–87.
[41] C.S. Peskin, Acta Numer. 11 (2002) 479–517.
[42] W.K. Liu, S. Jun, Y.F. Zhang, Internat. J. Numer. Methods Fluids 20 (8–9) (1995) 1081–1106.
[43] J.-S. Chen, C. Pan, C.-T. Wu, W.K. Liu, Comput. Methods Appl. Mech. Engrg. 139 (1–4) (1996) 195–227.
[44] Z. Fan, F. Qiu, A. Kaufman, S. Yoakum-Stover, Proceedings of the 2004 ACM/IEEE Conference on Supercomputing, IEEE Computer Society, 2004, p. 47.
[45] M. Schulz, M. Krafczyk, J. Tölke, E. Rank, High Performance Scientific and Engineering Computing, Springer, 2002, pp. 115–122.
[46] M.D. Lindemer, S.G. Advani, A.K. Prasad, Comput. Model. Eng. Sci. 117 (3) (2018) 527–553.
[47] S. Succi, The Lattice Boltzmann Equation: For Fluid Dynamics and beyond, Oxford university press, 2001.
[48] Z.-G. Feng, E.E. Michaelides, J. Comput. Phys. 195 (2) (2004) 602–628.
[49] S. Chen, G.D. Doolen, Annu. Rev. Fluid Mech. 30 (1) (1998) 329–364.
[50] Y. Qian, D. d'Humières, P. Lallemand, Europhys. Lett. 17 (6) (1992) 479.
[51] C.K. Aidun, J.R. Clausen, Annu. Rev. Fluid Mech. 42 (2010) 439–472.
[52] J. Latt, Palabos, Parallel Lattice Boltzmann Solver, FlowKit, Lausanne, Switzerland, 2009.
[53] Z. Liu, H. Wu, Appl. Therm. Eng. 93 (2016) 1394–1402.
[54] Y. Jin, M. Uth, H. Herwig, Comput. Fluids 107 (2015) 77–88.
[55] J. Tan, T.R. Sinno, S.L. Diamond, J. Comput. Sci. 25 (2018) 89–100.
[56] X. Wang, L.T. Zhang, Comput. Methods Appl. Mech. Engrg. 267 (2013) 150–169.
[57] H. Krüger, Computer Simulation Study of Collective Phenomena in Dense Suspensions of Red Blood Cells under Shear, Springer Science & Business Media, 2012.
[58] Y. Sui, Y.-T. Chew, P. Roy, H.-T. Low, J. Comput. Phys. 227 (12) (2008) 6351–6371.
[59] R. Macmeccan, J. Clausen, G. Neitzel, C. Aidun, J. Fluid Mech. 618 (2009) 13.
[60] L.T. Zhang, Int. J. Comput. Methods 14 (06) (2017) 1750068.
[61] M. Ostoja-Starzewski, Appl. Mech. Rev. 55 (1) (2002) 35–60.
[62] A.A. Gusev, Phys. Rev. Lett. 93 (3) (2004) 034302.
[63] H. Laubie, F. Radjaï, R. Pellenq, F.-J. Ulm, J. Mech. Phys. Solids 105 (2017) 116–130.
[64] G.-F. Zhao, J. Fang, J. Zhao, Int. J. Numer. Anal. Methods Geomech. 35 (8) (2011) 859–885.
[65] E. Schlangen, E. Garboczi, Eng. Fract. Mech. 57 (2–3) (1997) 319–332.
[66] Z.P. Bažant, M.R. Tabbara, M.T. Kazemi, G. Pijaudier-Cabot, J. Eng. Mech. 116 (8) (1990) 1686–1705.
[67] V.V. Yashin, A.C. Balazs, Science 314 (5800) (2006) 798–801.
[68] V.V. Yashin, A.C. Balazs, J. Chem. Phys. 126 (12) (2007) 124707.
[69] T. Zhang, Extreme Mech. Lett. 26 (2019) 40–45.
[70] F. Mackay, S.T. Ollila, C. Denniston, Comput. Phys. Comm. 184 (8) (2013) 2021–2031.
[71] Z. Guo, C. Zheng, B. Shi, Phys. Rev. E 65 (4) (2002) 046308.
[72] Q. Zou, X. He, Phys. Fluids 9 (6) (1997) 1591–1598.

[73] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, E. Viggen, The Lattice Boltzmann Method: Principles and Practice, Springer, Berlin, 2016.
[74] Y. Li, M. Kröger, Carbon 50 (5) (2012) 1793–1806.
[75] C. Li, T.-W. Chou, Int. J. Solids Struct. 40 (10) (2003) 2487–2499.
[76] J. Zhao, J.-W. Jiang, L. Wang, W. Guo, T. Rabczuk, J. Mech. Phys. Solids 71 (2014) 197–218.
[77] R.W. Ogden, Non-Linear Elastic Deformations, Courier Corporation, 1997.
[78] T. Belytschko, W.K. Liu, B. Moran, K. Elkhodary, Nonlinear Finite Elements for Continua and Structures, John wiley & sons, 2013.
[79] E. de Souza Neto, D. Perić, M. Dutko, D. Owen, Int. J. Solids Struct. 33 (20–22) (1996) 3277–3296.
[80] A. Gent, Rubber Chem. Technol. 69 (1) (1996) 59–61.
[81] E.M. Arruda, M.C. Boyce, J. Mech. Phys. Solids 41 (2) (1993) 389–412.
[82] D.A. Fedosov, B. Caswell, G.E. Karniadakis, Biophys. J. 98 (10) (2010) 2215–2225.
[83] A. Yazdani, P. Bagchi, J. Fluid Mech. 718 (2013) 569–595.
[84] T. Krüger, F. Varnik, D. Raabe, Comput. Math. Appl. 61 (12) (2011) 3485–3505.
[85] D. Barthès-Biesel, C. R. Phys. 10 (8) (2009) 764–774.
[86] A. Farutin, T. Biben, C. Misbah, J. Comput. Phys. 275 (2014) 539–568.
[87] I.V. Pivkin, G.E. Karniadakis, Phys. Rev. Lett. 101 (11) (2008) 118105.
[88] J. Tan, A. Thomas, Y. Liu, Soft Matter 8 (6) (2012) 1934–1946.
[89] M.P. Allen, D.J. Tildesley, Computer Simulation of Liquids, Oxford university press, 1989.
[90] M. Dao, J. Li, S. Suresh, Mater. Sci. Eng. C 26 (8) (2006) 1232–1244.
[91] Y.-H. Tang, S. Kudo, X. Bian, Z. Li, G.E. Karniadakis, J. Comput. Phys. 297 (2015) 13–31.
[92] Y. Wang, Z. Li, J. Xu, C. Yang, G.E. Karniadakis, Soft Matter 15 (8) (2019) 1747–1757.
[93] L. Zhang, X. Wu, IEEE Trans. Image Process. 15 (8) (2006) 2226–2238.
[94] W.K. Liu, Y. Chen, S. Jun, J. Chen, T. Belytschko, C. Pan, R. Uras, C. Chang, Arch. Comput. Methods Eng. 3 (1) (1996) 3–80.
[95] J.F. Hughes, A. Van Dam, J.D. Foley, M. McGuire, S.K. Feiner, D.F. Sklar, Computer Graphics: Principles and Practice, Pearson Education, 2014.
[96] M. Lindenbaum, S. Markovitch, D. Rusakov, Mach. Learn. 54 (2) (2004) 125–152.
[97] W.-K. Liu, S. Li, T. Belytschko, Comput. Methods Appl. Mech. Engrg. 143 (1–2) (1997) 113–154.
[98] E.H. Dowell, K.C. Hall, Annu. Rev. Fluid Mech. 33 (1) (2001) 445–490.
[99] Y. Bazilevs, K. Takizawa, T.E. Tezduyar, Computational Fluid-Structure Interaction: Methods and Applications, John Wiley & Sons, 2013.
[100] Y. Kim, C.S. Peskin, Phys. Fluids 19 (5) (2007) 053103.
[101] Hibbett, Karlsson, Sorensen, ABAQUS/Standard: User's Manual, Vol. 1, Hibbitt, Karlsson & Sorensen, 1998.
[102] S. Turek, J. Hron, M. Razzaq, H. Wobker, M. Schäfer, Fluid Structure Interaction II, Springer, 2011, pp. 413–424.
[103] Z. Lin, A. Hess, Z. Yu, S. Cai, T. Gao, J. Comput. Phys. 376 (2019) 1138–1155.
[104] P.-O. Persson, Mesh Generation for Implicit Geometries (Ph.D. thesis), Massachusetts Institute of Technology, 2005.
[105] E. Lac, D. Barthes-Biesel, N. Pelekasis, J. Tsamopoulos, J. Fluid Mech. 516 (2004) 303–334.
[106] J.D. Anderson Jr., Fundamentals of Aerodynamics, Tata McGraw-Hill Education, 2010.
[107] C. Duprat, H.A. Shore, Fluid-Structure Interactions in Low-Reynolds-Number Flows, Royal Society of Chemistry, 2015.
[108] E. Lauga, Phys. Rev. E 75 (4) (2007) 041916.
[109] S. Plimpton, J. Comput. Phys. 117 (1) (1995) 1–19.